A Project Report on
Music Streaming Service

HERTZ

DEVELOPED BY:

IT034 Soumyadeep Ghosh
IT038 Niken Goswami
IT042 Dishank Inani
IT044 Naamsukh Jobanputra

Guided By
Internal Guide:
Prof. Archana N. Vyas

Department of Information Technology
Faculty of Technology
DD University

Department of Information Technology Faculty of Technology,
Dharmsinh Desai University College Road, Nadiad-387001
October-2021

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled "Music Streaming system" is a bonafide report of the work carried out by

1) **Soumyadeep Ghosh** Student ID No: **19ITUOS094**
2) **Niken Goswami** Student ID No: **19ITUBS059**
3) **Naamsukh Jobanputra** Student ID No: **19ITUOS118**
4) **Dishank Inani** Student ID No: **19ITUOS107**

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during the academic year 2021-2022.

Prof. Archana N. Vyas
Project Guide, Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date: 21/10/2021

Prof. Vipul Dabhi
Head, Department of Information Technology

COMMENDATION

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of our project "Hertz."

The success and ultimate conclusion of this project necessitated a great deal of advice and support from a large number of individuals, and we are incredibly fortunate to have received it all along with the project's completion.

We owe a debt of appreciation to Prof. Archana N. Vyas, our project guide, who took an interest in our project work and directed us through it till it was completed by giving all of the required assistance for creating a solid Database System.

We'd also want to express our gratitude to all of our speakers. Finally, we express our gratitude to all of our friends and colleagues.

INDEX

Source Code - https://github.com/orgs/AOTitans/repositories

1. SYSTEM OVERVIEW

1.1 CURRENT SYSTEM

Our database will be designed like the Spotify streaming service, with artists, songs, categories, users, user playlists, and so on. Our major focus will be on structuring this data in such a way that playlists can be stored and a recommendation system for our users can be implemented later. It will also be done in an effective manner since we do not want our database to become redundant.

Within our platform, we want to encourage users to communicate with one another. The development of a 'collaboration' and 'following' connection inside our database addresses this aim.

1.2 OBJECTIVES OF THE PROPOSED SYSTEM

Music has progressed from being limited to tangible media to being widely available through digital methods during the last two decades. As a result, the method of music purchase has changed, moving from a pay-per-song approach to a more contemporary streaming strategy.

Our Project intends to address these two trends while also enhancing the user experience by offering tailored suggestions, a social network, and 24/7 access with the option to listen to music on the move.

1.3 ADVANTAGES OF THE PROPOSED SYSTEM

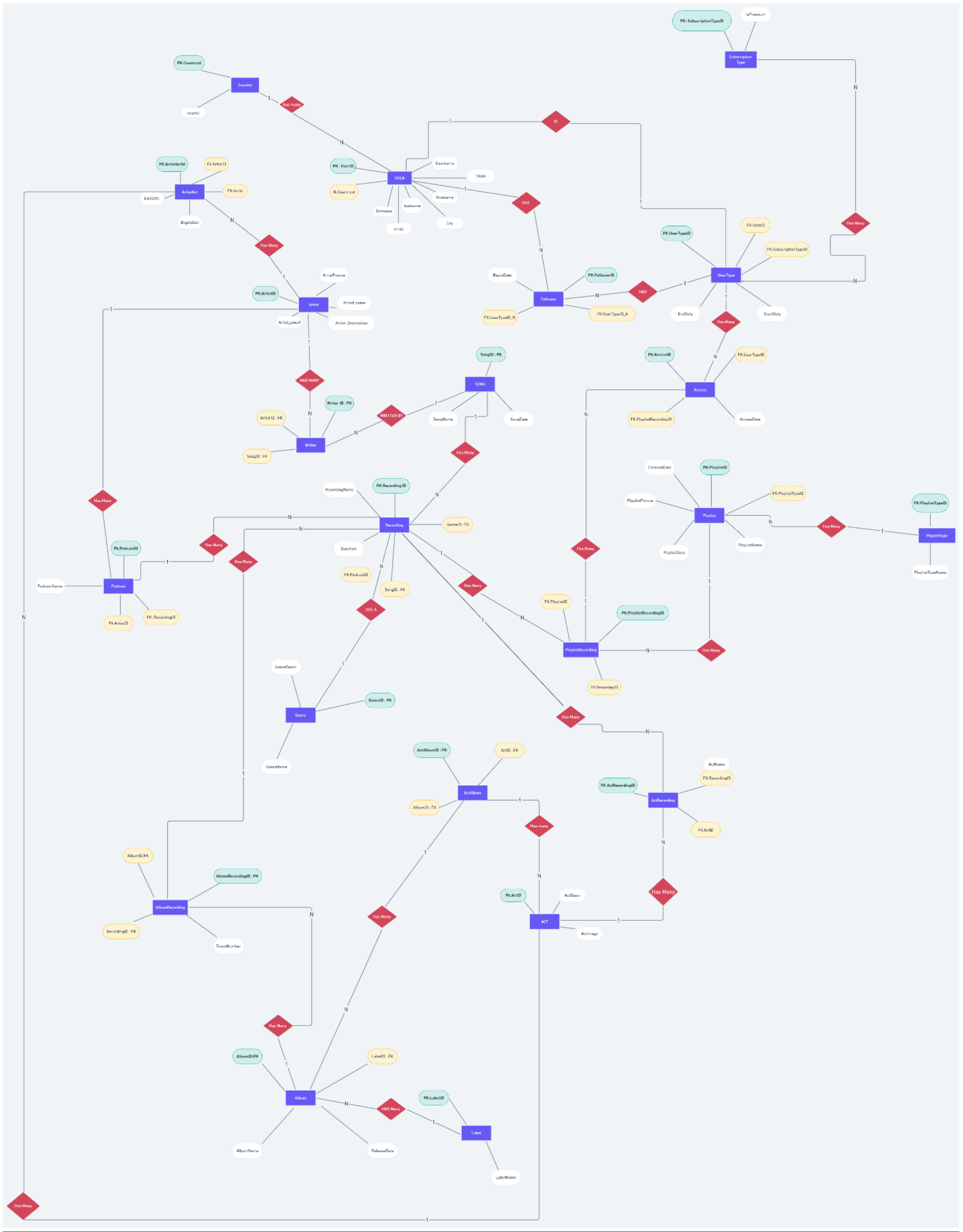
- Hertz is a music-playing application in which users can listen to their favourite songs as well as create personalized playlists as well as listen to the ones created by our developers.

- Hertz looks forward to bringing podcasts to a new level as well as giving artists a smooth platform to record their songs and podcasts.

- This app has a highly normalized database so as to make efficient access of data and reduce data redundancy.

We tend to make this application more personalized.

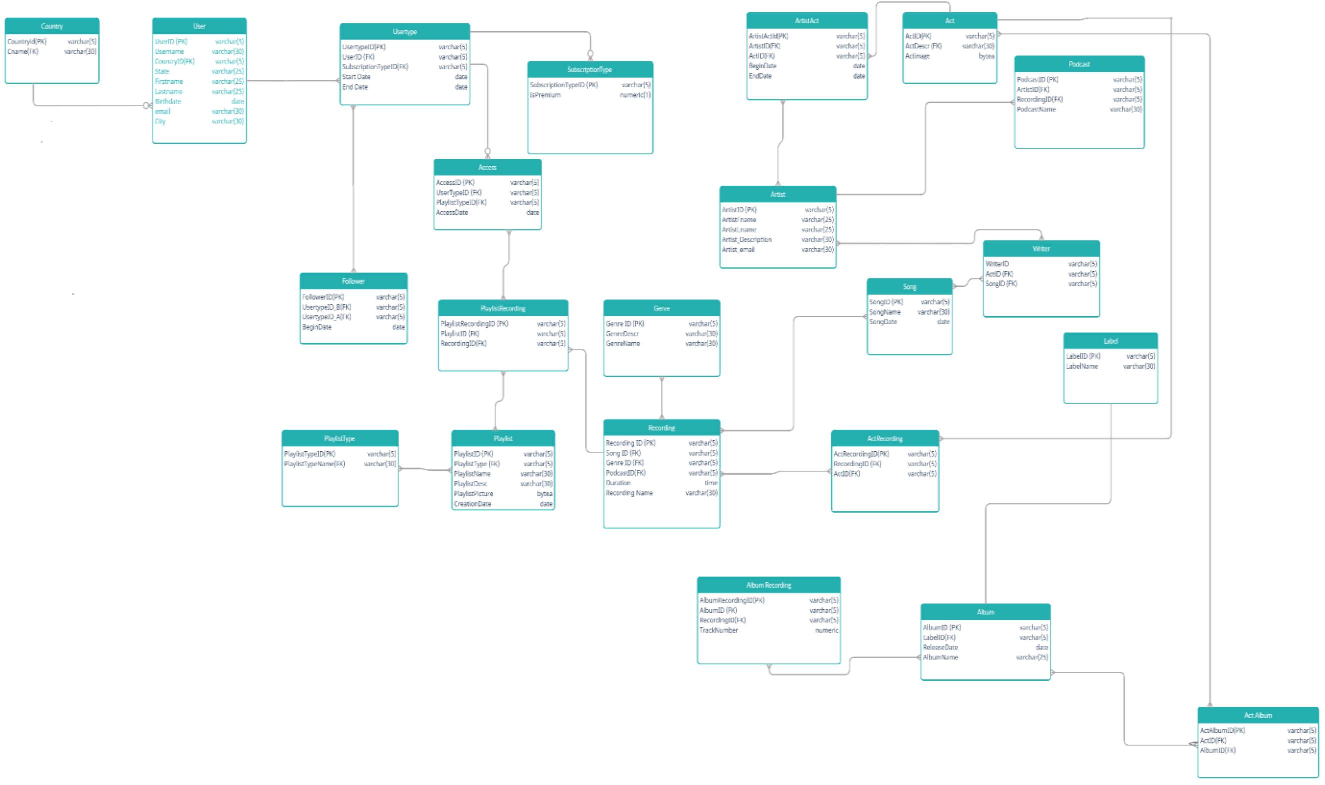
2. ENTITY-RELATIONSHIP MODEL



Link - https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD

Password :12345

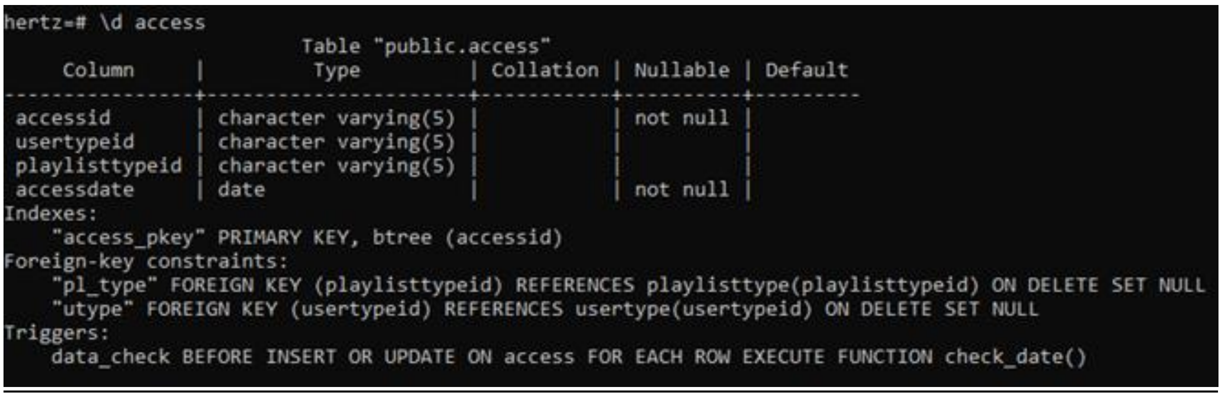
3. RELATIONAL SCHEMA



Link - https://app.creately.com/diagram/yQAR0D8Dgpa/edit

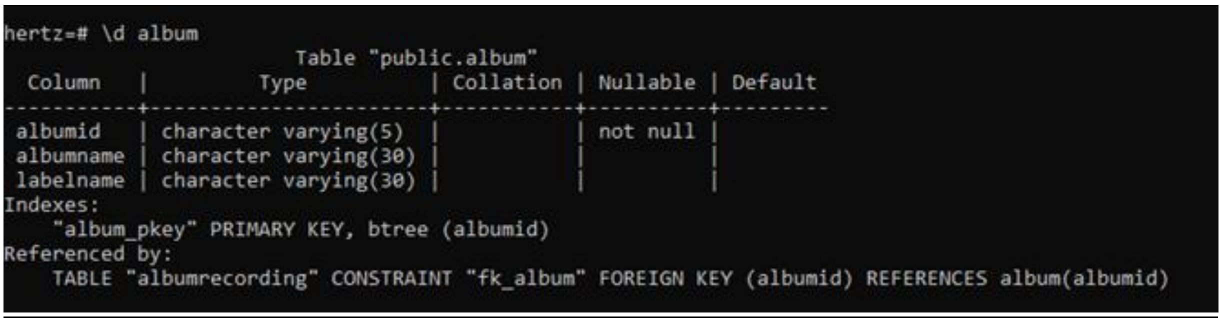
4.DATA DICTIONARY

4.1 access



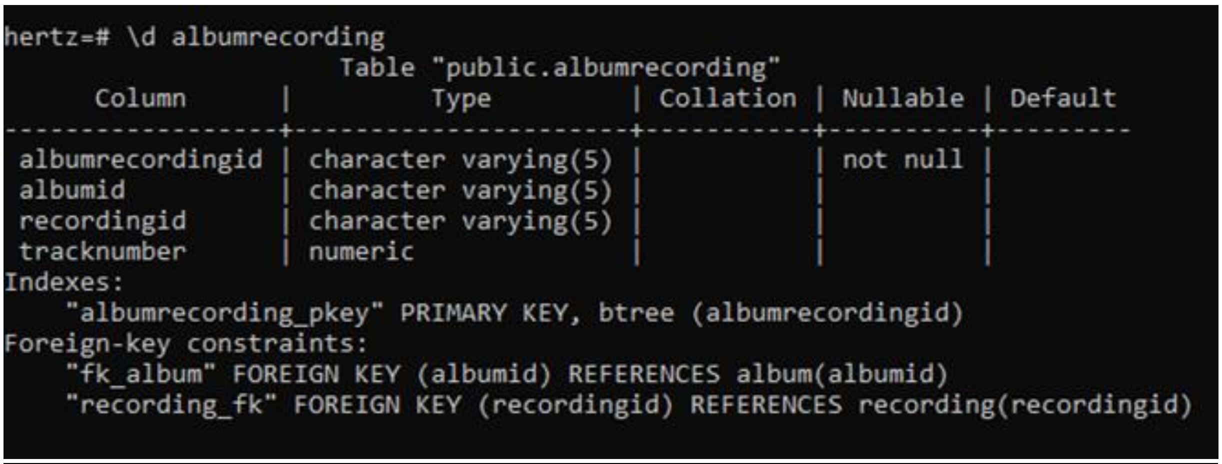
```
hertz=# \d access
                       Table "public.access"
     Column     |         Type          | Collation | Nullable | Default
----------------+-----------------------+-----------+----------+---------
 accessid       | character varying(5)  |           | not null |
 usertypeid     | character varying(5)  |           |          |
 playlisttypeid | character varying(5)  |           |          |
 accessdate     | date                  |           | not null |
Indexes:
    "access_pkey" PRIMARY KEY, btree (accessid)
Foreign-key constraints:
    "pl_type" FOREIGN KEY (playlisttypeid) REFERENCES playlisttype(playlisttypeid) ON DELETE SET NULL
    "utype" FOREIGN KEY (usertypeid) REFERENCES usertype(usertypeid) ON DELETE SET NULL
Triggers:
    data_check BEFORE INSERT OR UPDATE ON access FOR EACH ROW EXECUTE FUNCTION check_date()
```

4.2 album



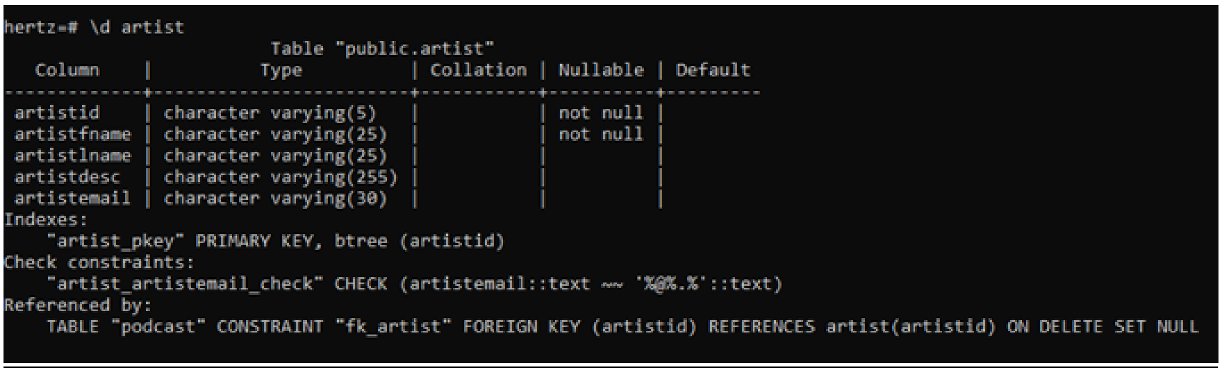
```
hertz=# \d album
                       Table "public.album"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 albumid   | character varying(5)  |           | not null |
 albumname | character varying(30) |           |          |
 labelname | character varying(30) |           |          |
Indexes:
    "album_pkey" PRIMARY KEY, btree (albumid)
Referenced by:
    TABLE "albumrecording" CONSTRAINT "fk_album" FOREIGN KEY (albumid) REFERENCES album(albumid)
```

4.3 album recording



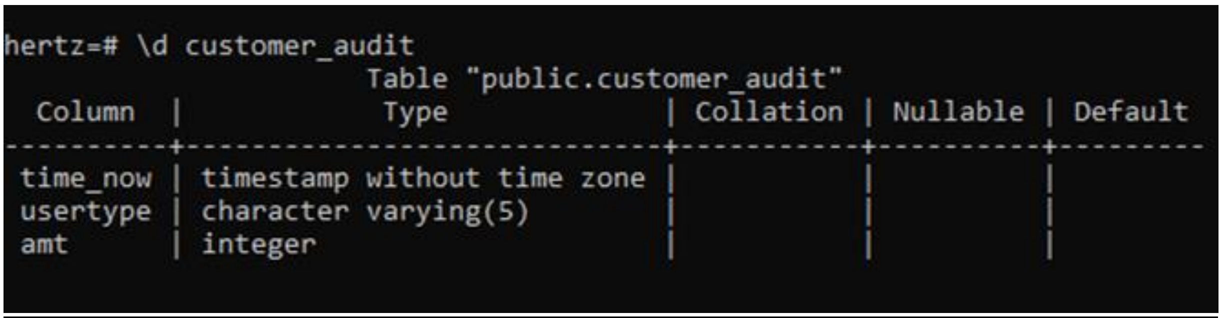
```
hertz=# \d albumrecording
                    Table "public.albumrecording"
      Column      |         Type          | Collation | Nullable | Default
------------------+-----------------------+-----------+----------+---------
 albumrecordingid | character varying(5)  |           | not null |
 albumid          | character varying(5)  |           |          |
 recordingid      | character varying(5)  |           |          |
 tracknumber      | numeric               |           |          |
Indexes:
    "albumrecording_pkey" PRIMARY KEY, btree (albumrecordingid)
Foreign-key constraints:
    "fk_album" FOREIGN KEY (albumid) REFERENCES album(albumid)
    "recording_fk" FOREIGN KEY (recordingid) REFERENCES recording(recordingid)
```

4.4 artist



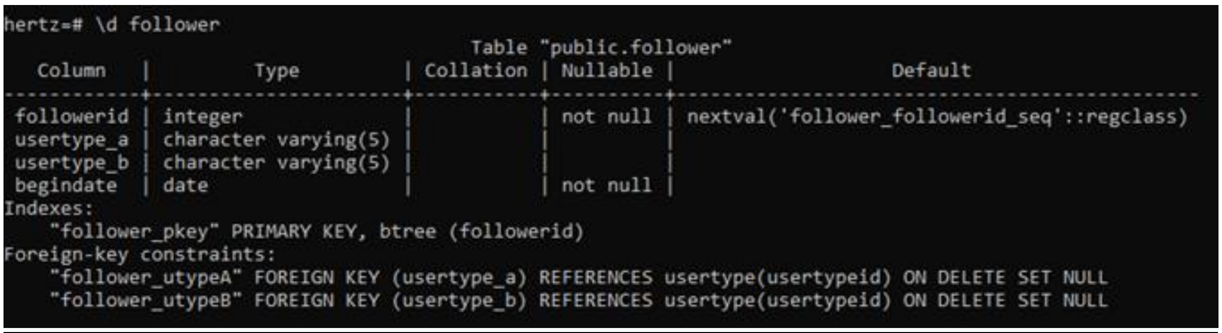
```
hertz=# \d artist
                         Table "public.artist"
   Column    |          Type          | Collation | Nullable | Default
-------------+------------------------+-----------+----------+---------
 artistid    | character varying(5)   |           | not null |
 artistfname | character varying(25)  |           | not null |
 artistlname | character varying(25)  |           |          |
 artistdesc  | character varying(255) |           |          |
 artistemail | character varying(30)  |           |          |
Indexes:
    "artist_pkey" PRIMARY KEY, btree (artistid)
Check constraints:
    "artist_artistemail_check" CHECK (artistemail::text ~~ '%@%.%'::text)
Referenced by:
    TABLE "podcast" CONSTRAINT "fk_artist" FOREIGN KEY (artistid) REFERENCES artist(artistid) ON DELETE SET NULL
```

4.5 customer audit



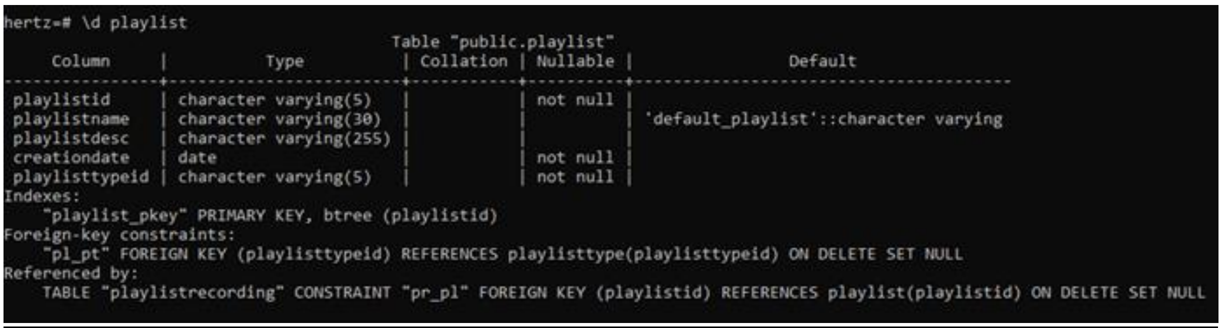
```
hertz=# \d customer_audit
                   Table "public.customer_audit"
  Column   |            Type             | Collation | Nullable | Default
-----------+-----------------------------+-----------+----------+---------
 time_now  | timestamp without time zone |           |          |
 usertype  | character varying(5)        |           |          |
 amt       | integer                     |           |          |
```

4.6 follower



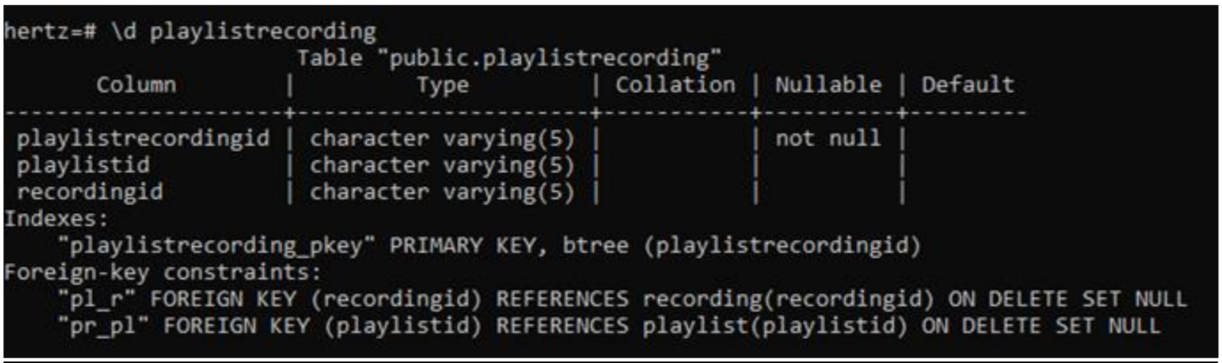
```
hertz=# \d follower
                                      Table "public.follower"
   Column   |         Type         | Collation | Nullable |                  Default
------------+----------------------+-----------+----------+-------------------------------------------
 followerid | integer              |           | not null | nextval('follower_followerid_seq'::regclass)
 usertype_a | character varying(5) |           |          |
 usertype_b | character varying(5) |           |          |
 begindate  | date                 |           | not null |
Indexes:
    "follower_pkey" PRIMARY KEY, btree (followerid)
Foreign-key constraints:
    "follower_utypeA" FOREIGN KEY (usertype_a) REFERENCES usertype(usertypeid) ON DELETE SET NULL
    "follower_utypeB" FOREIGN KEY (usertype_b) REFERENCES usertype(usertypeid) ON DELETE SET NULL
```

4.7 playlist



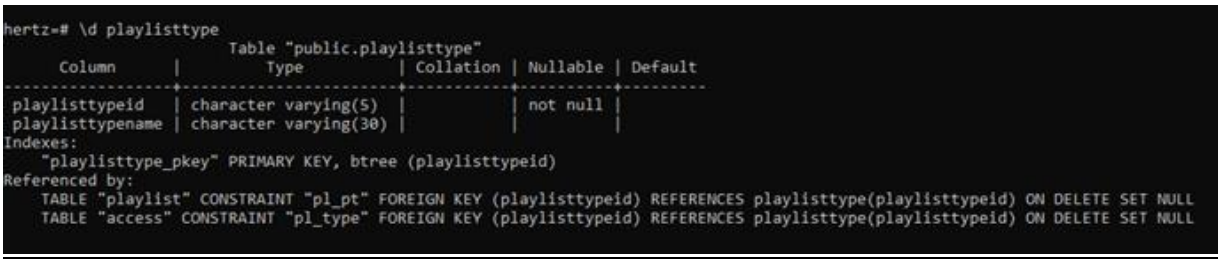
```
hertz=# \d playlist
                                         Table "public.playlist"
    Column     |          Type          | Collation | Nullable |                Default
---------------+------------------------+-----------+----------+---------------------------------------
 playlistid     | character varying(5)   |           | not null |
 playlistname   | character varying(30)  |           |          | 'default_playlist'::character varying
 playlistdesc   | character varying(255) |           |          |
 creationdate   | date                   |           | not null |
 playlisttypeid | character varying(5)   |           | not null |
Indexes:
    "playlist_pkey" PRIMARY KEY, btree (playlistid)
Foreign-key constraints:
    "pl_pt" FOREIGN KEY (playlisttypeid) REFERENCES playlisttype(playlisttypeid) ON DELETE SET NULL
Referenced by:
    TABLE "playlistrecording" CONSTRAINT "pr_pl" FOREIGN KEY (playlistid) REFERENCES playlist(playlistid) ON DELETE SET NULL
```

4.8 playlist recording



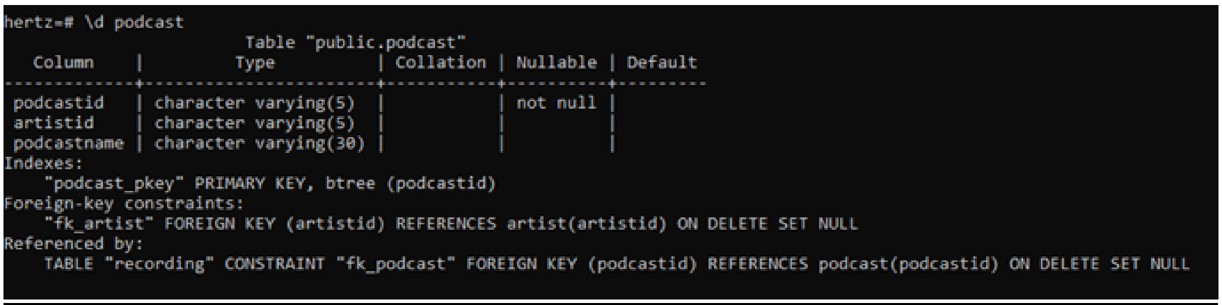
```
hertz=# \d playlistrecording
                       Table "public.playlistrecording"
       Column        |         Type         | Collation | Nullable | Default
---------------------+----------------------+-----------+----------+---------
 playlistrecordingid | character varying(5) |           | not null |
 playlistid          | character varying(5) |           |          |
 recordingid         | character varying(5) |           |          |
Indexes:
    "playlistrecording_pkey" PRIMARY KEY, btree (playlistrecordingid)
Foreign-key constraints:
    "pl_r" FOREIGN KEY (recordingid) REFERENCES recording(recordingid) ON DELETE SET NULL
    "pr_pl" FOREIGN KEY (playlistid) REFERENCES playlist(playlistid) ON DELETE SET NULL
```

4.9 playlisttype



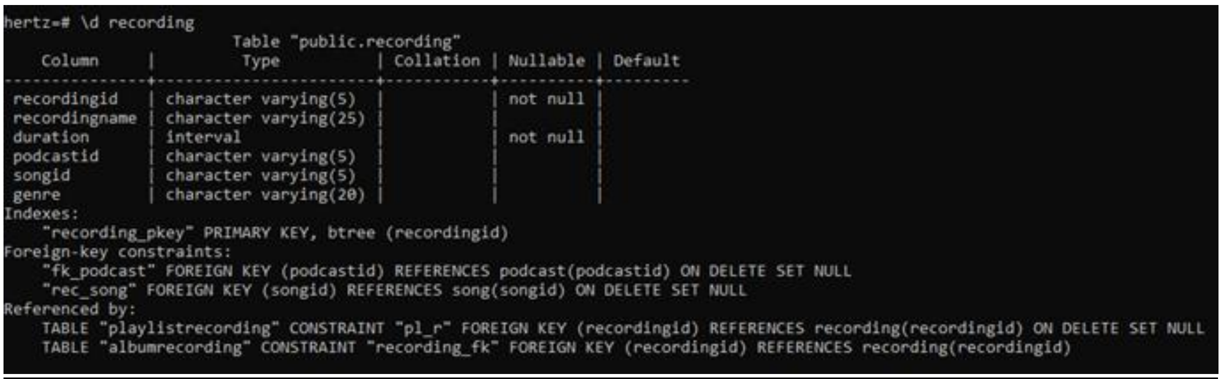
```
hertz=# \d playlisttype
                      Table "public.playlisttype"
      Column      |         Type          | Collation | Nullable | Default
------------------+-----------------------+-----------+----------+---------
 playlisttypeid   | character varying(5)  |           | not null |
 playlisttypename | character varying(30) |           |          |
Indexes:
    "playlisttype_pkey" PRIMARY KEY, btree (playlisttypeid)
Referenced by:
    TABLE "playlist" CONSTRAINT "pl_pt" FOREIGN KEY (playlisttypeid) REFERENCES playlisttype(playlisttypeid) ON DELETE SET NULL
    TABLE "access" CONSTRAINT "pl_type" FOREIGN KEY (playlisttypeid) REFERENCES playlisttype(playlisttypeid) ON DELETE SET NULL
```

4.10 podcast



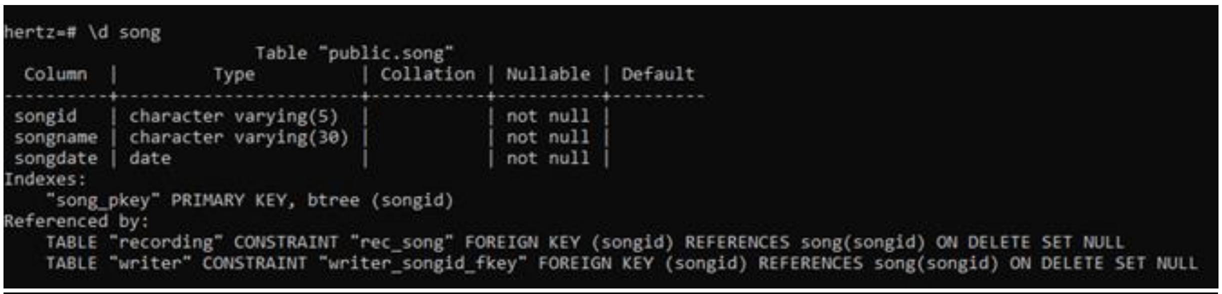
```
hertz=# \d podcast
                       Table "public.podcast"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 podcastid   | character varying(5)  |           | not null |
 artistid    | character varying(5)  |           |          |
 podcastname | character varying(30) |           |          |
Indexes:
    "podcast_pkey" PRIMARY KEY, btree (podcastid)
Foreign-key constraints:
    "fk_artist" FOREIGN KEY (artistid) REFERENCES artist(artistid) ON DELETE SET NULL
Referenced by:
    TABLE "recording" CONSTRAINT "fk_podcast" FOREIGN KEY (podcastid) REFERENCES podcast(podcastid) ON DELETE SET NULL
```

4.11 recording



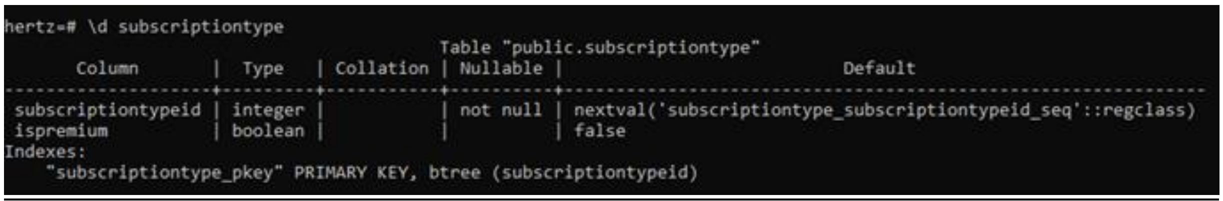
```
hertz=# \d recording
                       Table "public.recording"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 recordingid   | character varying(5)  |           | not null |
 recordingname | character varying(25) |           |          |
 duration      | interval              |           | not null |
 podcastid     | character varying(5)  |           |          |
 songid        | character varying(5)  |           |          |
 genre         | character varying(20) |           |          |
Indexes:
    "recording_pkey" PRIMARY KEY, btree (recordingid)
Foreign-key constraints:
    "fk_podcast" FOREIGN KEY (podcastid) REFERENCES podcast(podcastid) ON DELETE SET NULL
    "rec_song" FOREIGN KEY (songid) REFERENCES song(songid) ON DELETE SET NULL
Referenced by:
    TABLE "playlistrecording" CONSTRAINT "pl_r" FOREIGN KEY (recordingid) REFERENCES recording(recordingid) ON DELETE SET NULL
    TABLE "albumrecording" CONSTRAINT "recording_fk" FOREIGN KEY (recordingid) REFERENCES recording(recordingid)
```

4.12 song



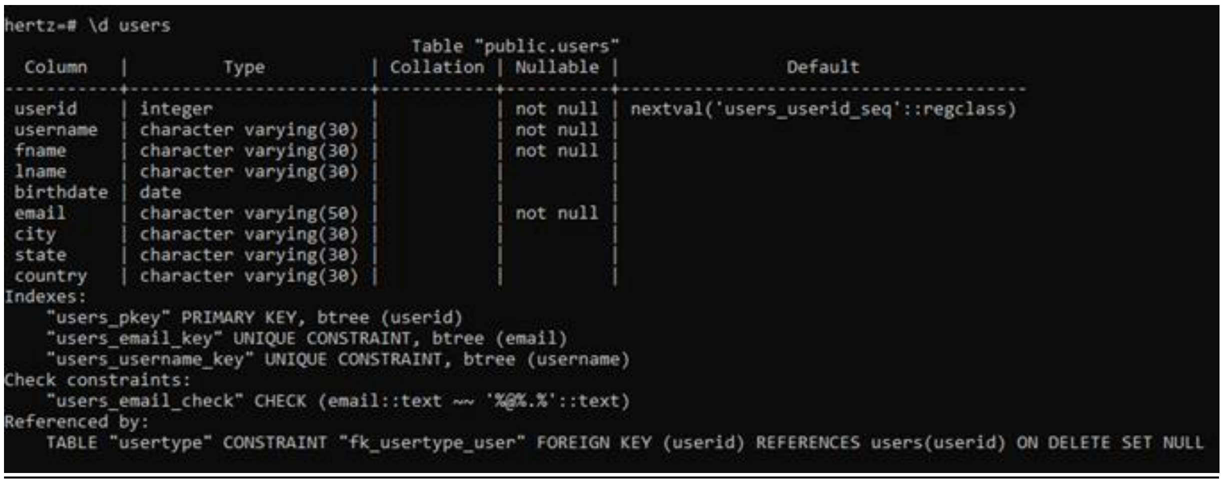
```
hertz=# \d song
                       Table "public.song"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 songid   | character varying(5)  |           | not null |
 songname | character varying(30) |           | not null |
 songdate | date                  |           | not null |
Indexes:
    "song_pkey" PRIMARY KEY, btree (songid)
Referenced by:
    TABLE "recording" CONSTRAINT "rec_song" FOREIGN KEY (songid) REFERENCES song(songid) ON DELETE SET NULL
    TABLE "writer" CONSTRAINT "writer_songid_fkey" FOREIGN KEY (songid) REFERENCES song(songid) ON DELETE SET NULL
```

4.13 subscription type



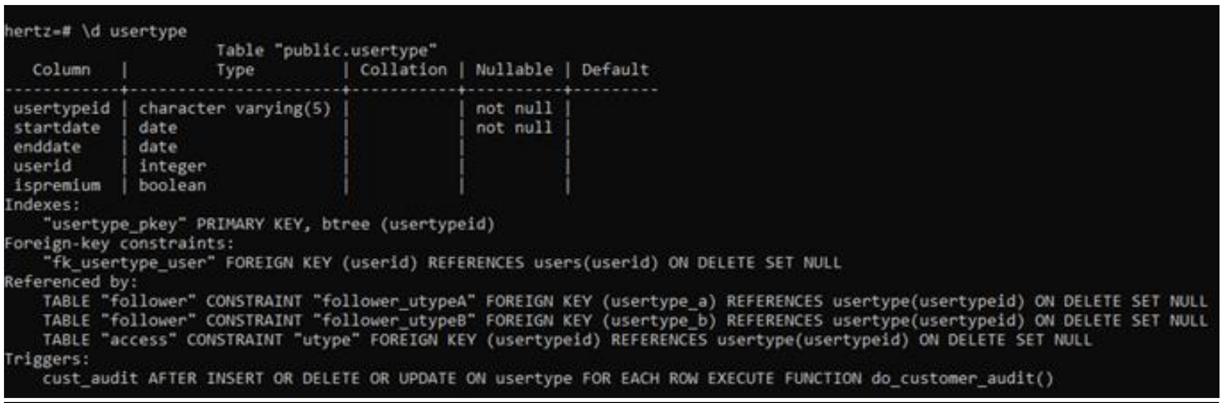
```
hertz=# \d subscriptiontype
                                   Table "public.subscriptiontype"
      Column       |  Type   | Collation | Nullable |                        Default
-------------------+---------+-----------+----------+-------------------------------------------------------------
 subscriptiontypeid | integer |           | not null | nextval('subscriptiontype_subscriptiontypeid_seq'::regclass)
 ispremium          | boolean |           |          | false
Indexes:
    "subscriptiontype_pkey" PRIMARY KEY, btree (subscriptiontypeid)
```

4.14 users



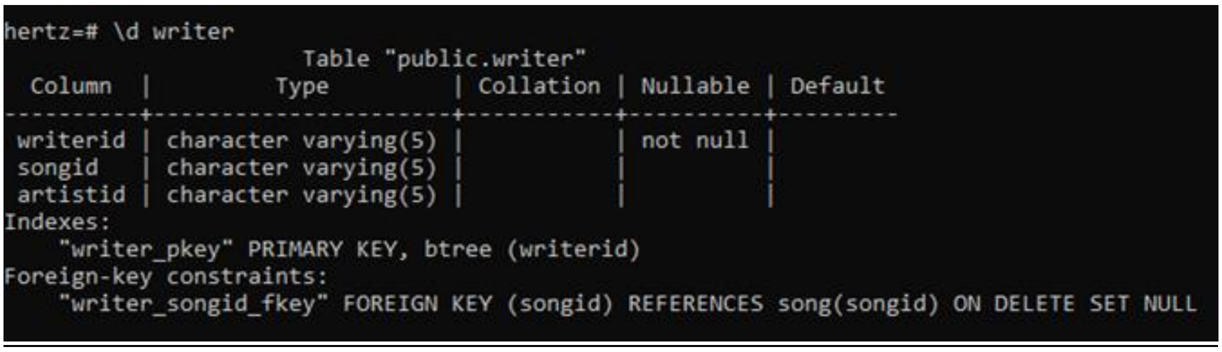
```
hertz=# \d users
                                       Table "public.users"
  Column   |         Type          | Collation | Nullable |              Default
-----------+-----------------------+-----------+----------+------------------------------------
 userid    | integer               |           | not null | nextval('users_userid_seq'::regclass)
 username  | character varying(30) |           | not null |
 fname     | character varying(30) |           | not null |
 lname     | character varying(30) |           |          |
 birthdate | date                  |           |          |
 email     | character varying(50) |           | not null |
 city      | character varying(30) |           |          |
 state     | character varying(30) |           |          |
 country   | character varying(30) |           |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (userid)
    "users_email_key" UNIQUE CONSTRAINT, btree (email)
    "users_username_key" UNIQUE CONSTRAINT, btree (username)
Check constraints:
    "users_email_check" CHECK (email::text ~~ '%@%.%'::text)
Referenced by:
    TABLE "usertype" CONSTRAINT "fk_usertype_user" FOREIGN KEY (userid) REFERENCES users(userid) ON DELETE SET NULL
```

4.15 usertype



```
hertz=# \d usertype
                      Table "public.usertype"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 usertypeid | character varying(5)  |           | not null |
 startdate  | date                  |           | not null |
 enddate    | date                  |           |          |
 userid     | integer               |           |          |
 ispremium  | boolean               |           |          |
Indexes:
    "usertype_pkey" PRIMARY KEY, btree (usertypeid)
Foreign-key constraints:
    "fk_usertype_user" FOREIGN KEY (userid) REFERENCES users(userid) ON DELETE SET NULL
Referenced by:
    TABLE "follower" CONSTRAINT "follower_utypeA" FOREIGN KEY (usertype_a) REFERENCES usertype(usertypeid) ON DELETE SET NULL
    TABLE "follower" CONSTRAINT "follower_utypeB" FOREIGN KEY (usertype_b) REFERENCES usertype(usertypeid) ON DELETE SET NULL
    TABLE "access" CONSTRAINT "utype" FOREIGN KEY (usertypeid) REFERENCES usertype(usertypeid) ON DELETE SET NULL
Triggers:
    cust_audit AFTER INSERT OR DELETE OR UPDATE ON usertype FOR EACH ROW EXECUTE FUNCTION do_customer_audit()
```

4.16 writer



```
hertz=# \d writer
                     Table "public.writer"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 writerid | character varying(5) |           | not null |
 songid   | character varying(5) |           |          |
 artistid | character varying(5) |           |          |
Indexes:
    "writer_pkey" PRIMARY KEY, btree (writerid)
Foreign-key constraints:
    "writer_songid_fkey" FOREIGN KEY (songid) REFERENCES song(songid) ON DELETE SET NULL
```

5. DATA IMPLEMENTATION

5.1 SCHEMA

5.1.1 access

```
create table access (accessid varchar(5) primary key,usertypeid varchar(5)
,playlisttypeid varchar(5),accessdate date not null, constraint "pl_type" foreign
key(playlisttypeid) references playlisttype(playlisttypeid) on delete set null, constraint
"utype" foreign key(usertypeid) references usertype(usertypeid) on delete set null);
```

5.1.2 album

```
create table album ( albumid varchar(5) primary key , albumname
varchar(30),labelname varchar(30));
```

5.1.3 albumrecording

```
create table albumrecording(albumrecordingid varchar(5) primary key,albumid
varchar(5),recordingid varchar(5),tracknumber numeric ,constraint "fk_album" foreign
key(albumid) references album(albumid),constraint "recording_fk" foreign
key(recordingid) references recording(recordingid));
```

5.1.4 artist

```
create table artist(artistid varchar(5) primary key,

artistfname varchar(25) not null,

artistlname varchar(25),

artistdesc varchar(255),

artistemail varchar(30));

alter table artist add constraint "artist_artistemail_check" check(artistemail like
'%@%.%');
```

5.1.5 customer_audit

```
create table customer_audit( time_now TIMESTAMP , usertype varchar(5) , amt
INT);
```

5.1.6 follower

```
create table follower(

followerid serial primary key,

usertype_A varchar(5),

usertype_B varchar(5),

begindate date not null,

constraint "follower_utypeA" foreign key (usertype_A)

references usertype(usertypeid) on delete set null,

constraint "follower_utypeB" foreign key (usertype_B)

references usertype(usertypeid) on delete set null);
```

5.1.7 playlist

```
create table playlist(

playlistid varchar(5) primary key,

playlistname varchar(30) default 'default_playlist',

playlistdesc varchar(255),

creationdate date not null,

playlisttypeid varchar(5) not null,

constraint "pl_pt" foreign key (playlisttypeid)

references playlisttype(playlisttypeid) on delete set null);
```

5.1.8 playlistrecording

create table playlistrecording (playlistrecordingid varchar(5) primary key ,playlistid varchar(5) , recordingid varchar(5) , constraint "pl_r" foreign key (recordingid) references recording(recordingid) on delete set null, constraint "pr_pl" foreign key (playlistid) references playlist(playlistid) on delete set null);

5.1.9 playlisttype

create table playlisttype(

playlisttypeid varchar(5) primary key,

playlisttypename varchar(30));

5.1.10 podcast

create table podcast(

podcastid varchar(5) primary key,

artistid varchar(5),

podcastname varchar(30),

constraint "fk_artist" foreign key (artistid)

references artist(artistid) on delete set null);

5.1.11 recording

create table recording(

recordingid varchar(5) primary key,

recordingname varchar(25),

duration interval not null,

podcastid varchar(5),

```
songid varchar(5),
genre varchar(20),
constraint "fk_podcast" foreign key (podcastid)
references podcast(podcastid) on delete set null,
constraint "rec_song" foreign key (songid)
references song(songid) on delete set null);
```

5.1.12 song

```
create table song(
songid varchar(5) primary key,
songname varchar(30) not null,
songdate date not null);
```

5.1.13 subscriptiontype

```
create table subscriptiontype(
subscriptiontypeid serial primary key,
ispremium boolean default false);
```

5.1.14 users

```
create table users(
userid serial primary key,
username varchar(30) not null,
fname varchar(30) not null,
lname varchar(30),
birthdate date,

email varchar(50) not null,

city varchar(30),

state varchar(30),

country varchar(30),

unique(email),

unique(username));


alter table users add constraint "users_email_check" check(email like '%@%.%');
```

5.1.15 usertype

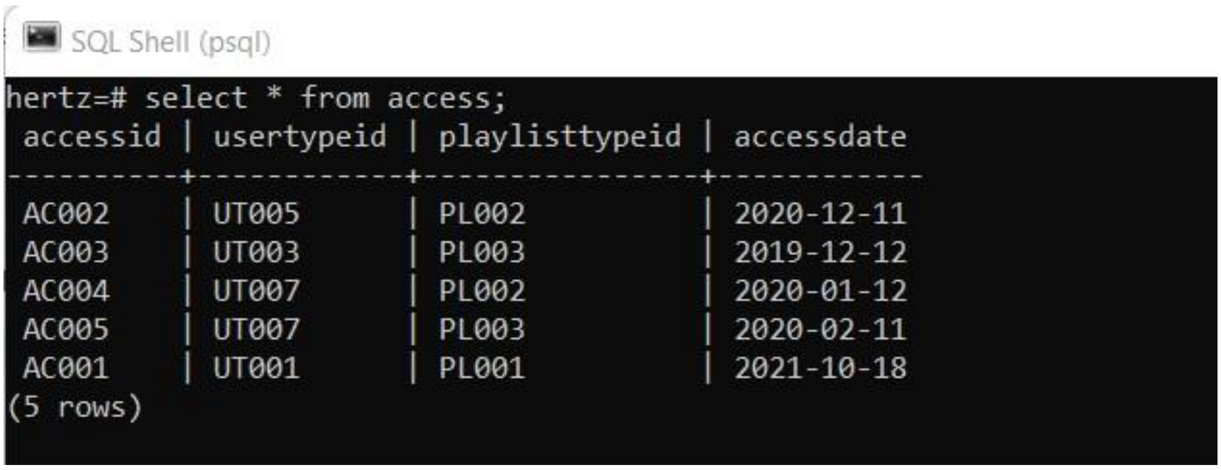
```
create table usertype(

usertypeid varchar(5) primary key,

startdate date not null,

enddate date,

userid integer,

ispremium boolean,

constraint "fk_usertype_user" foreign key (userid)

references users(userid) on delete set null);
```

5.1.16 writer

```
create table writer(writerid varchar(5) primary key not null , artistid varchar(5) , songid
varchar(5) ,constraint "writer_songid_fkey" foreign key(songid) references
song(songid) on delete set null);
```

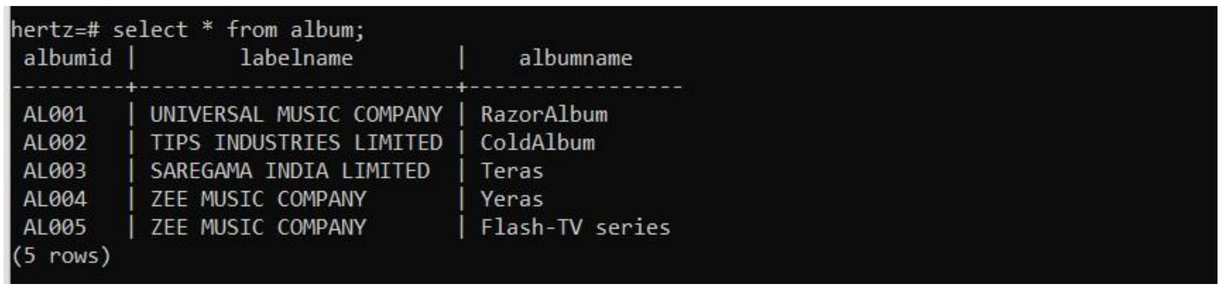
5.2 INSERTING DATA VALUES

5.2.1 Access:



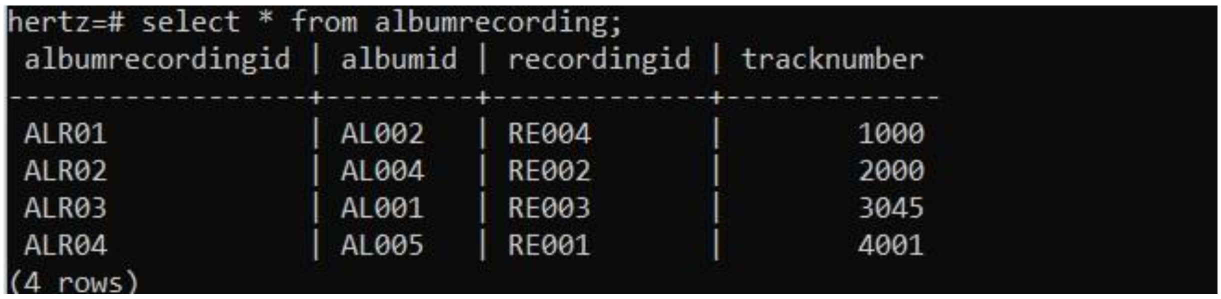
```
hertz=# select * from access;
 accessid | usertypeid | playlisttypeid | accessdate
----------+------------+----------------+------------
 AC002    | UT005      | PL002          | 2020-12-11
 AC003    | UT003      | PL003          | 2019-12-12
 AC004    | UT007      | PL002          | 2020-01-12
 AC005    | UT007      | PL003          | 2020-02-11
 AC001    | UT001      | PL001          | 2021-10-18
(5 rows)
```

5.2.2 Album :



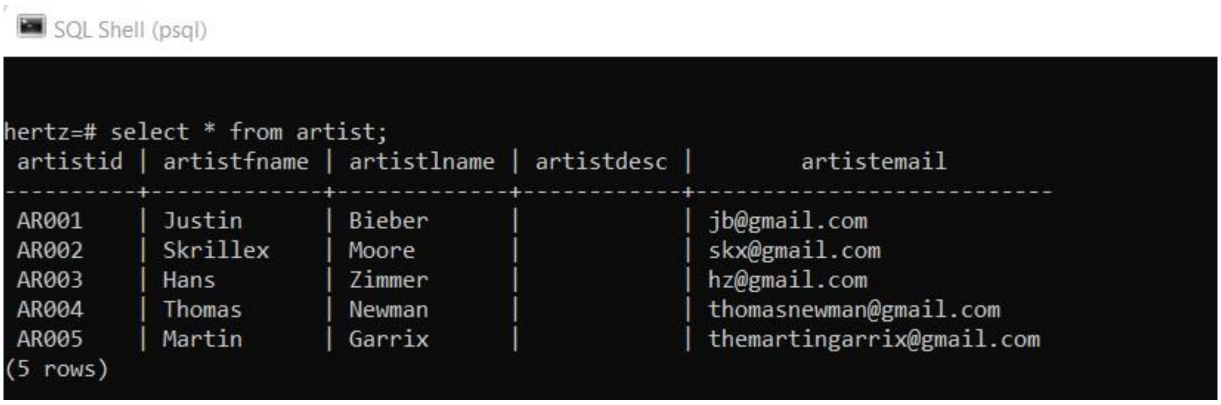
```
hertz=# select * from album;
 albumid |       labelname        |    albumname
---------+------------------------+----------------
 AL001   | UNIVERSAL MUSIC COMPANY | RazorAlbum
 AL002   | TIPS INDUSTRIES LIMITED | ColdAlbum
 AL003   | SAREGAMA INDIA LIMITED  | Teras
 AL004   | ZEE MUSIC COMPANY       | Yeras
 AL005   | ZEE MUSIC COMPANY       | Flash-TV series
(5 rows)
```

5.2.3 Album Recording :



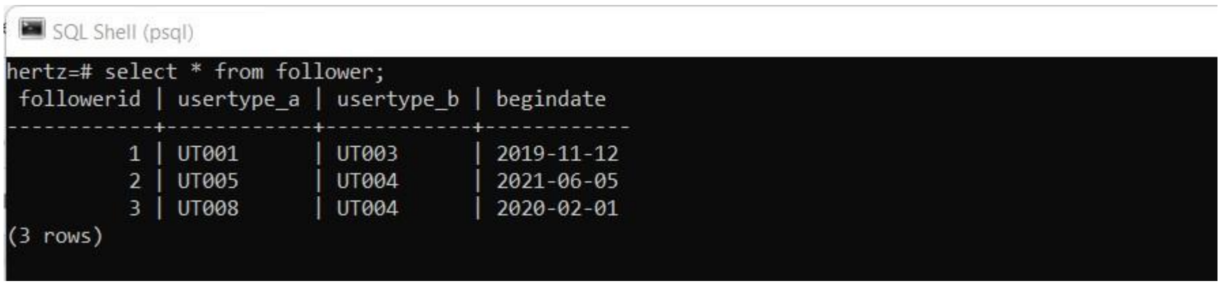
```
hertz=# select * from albumrecording;
 albumrecordingid | albumid | recordingid | tracknumber
------------------+---------+-------------+-------------
 ALR01            | AL002   | RE004       |        1000
 ALR02            | AL004   | RE002       |        2000
 ALR03            | AL001   | RE003       |        3045
 ALR04            | AL005   | RE001       |        4001
(4 rows)
```

5.2.4 Artist :


SQL Shell (psql)

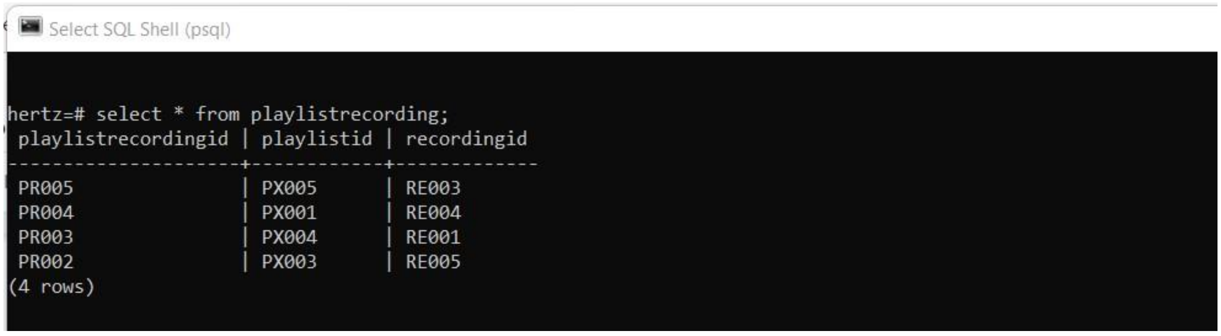
```
hertz=# select * from artist;
 artistid | artistfname | artistlname | artistdesc |        artistemail
----------+-------------+-------------+------------+---------------------------
 AR001    | Justin      | Bieber      |            | jb@gmail.com
 AR002    | Skrillex    | Moore       |            | skx@gmail.com
 AR003    | Hans        | Zimmer      |            | hz@gmail.com
 AR004    | Thomas      | Newman      |            | thomasnewman@gmail.com
 AR005    | Martin      | Garrix      |            | themartingarrix@gmail.com
(5 rows)
```

5.2.5 Follower :


SQL Shell (psql)

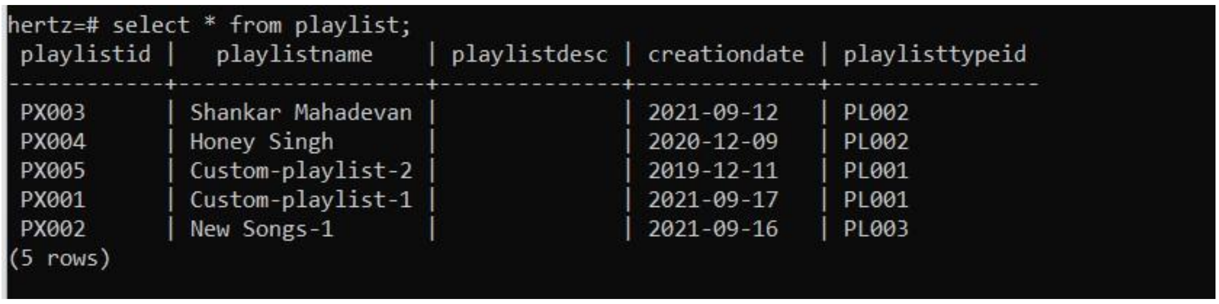
```
hertz=# select * from follower;
 followerid | usertype_a | usertype_b | begindate
------------+------------+------------+------------
          1 | UT001      | UT003      | 2019-11-12
          2 | UT005      | UT004      | 2021-06-05
          3 | UT008      | UT004      | 2020-02-01
(3 rows)
```

5.2.6 Playlist Recording :


Select SQL Shell (psql)

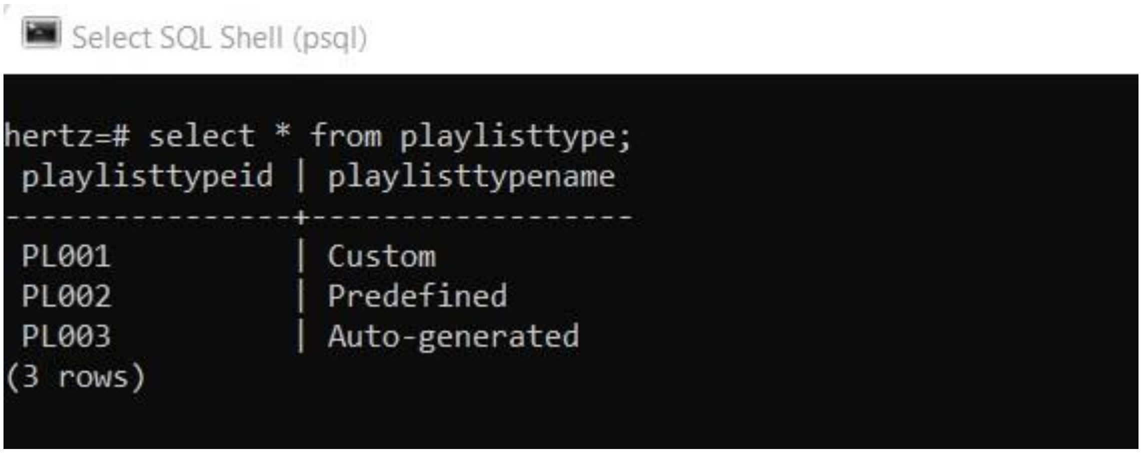
```
hertz=# select * from playlistrecording;
 playlistrecordingid | playlistid | recordingid
---------------------+------------+-------------
 PR005               | PX005      | RE003
 PR004               | PX001      | RE004
 PR003               | PX004      | RE001
 PR002               | PX003      | RE005
(4 rows)
```

5.2.7 Playlist :



```
hertz=# select * from playlist;
 playlistid |   playlistname    | playlistdesc | creationdate | playlisttypeid
------------+-------------------+--------------+--------------+----------------
 PX003      | Shankar Mahadevan |              | 2021-09-12   | PL002
 PX004      | Honey Singh       |              | 2020-12-09   | PL002
 PX005      | Custom-playlist-2 |              | 2019-12-11   | PL001
 PX001      | Custom-playlist-1 |              | 2021-09-17   | PL001
 PX002      | New Songs-1       |              | 2021-09-16   | PL003
(5 rows)
```

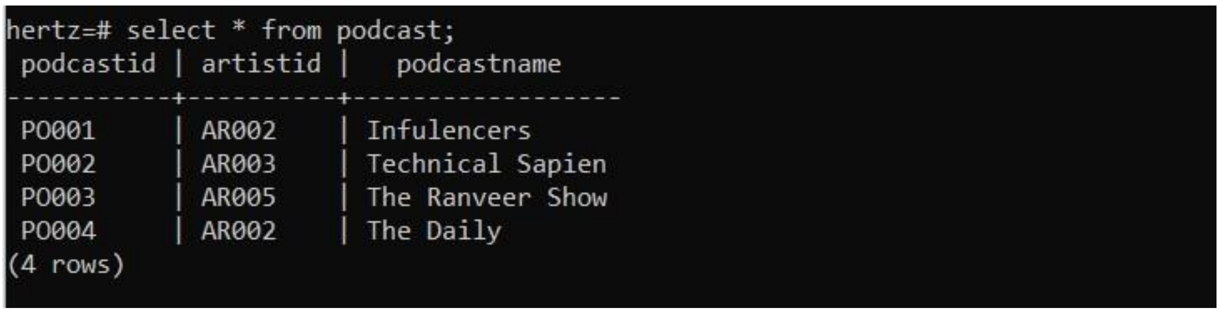
5.2.8 Playlist Type :



Select SQL Shell (psql)

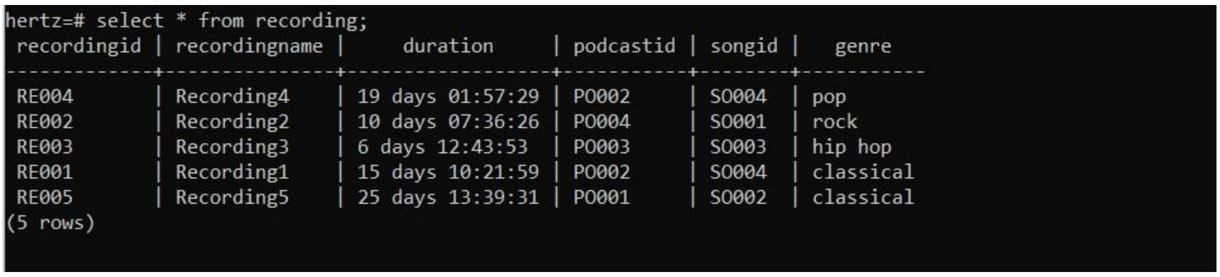
```
hertz=# select * from playlisttype;
 playlisttypeid | playlisttypename
----------------+------------------
 PL001          | Custom
 PL002          | Predefined
 PL003          | Auto-generated
(3 rows)
```

5.2.9 Podcast :



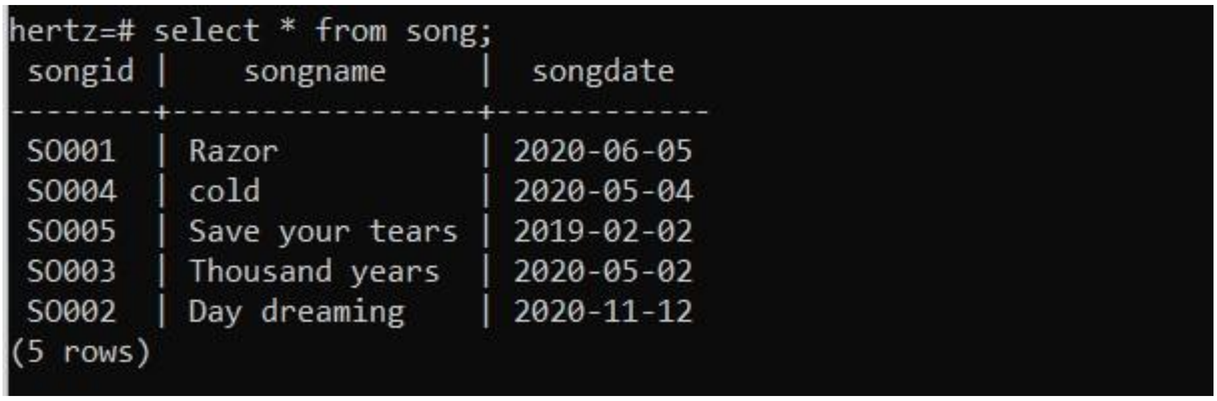
```
hertz=# select * from podcast;
 podcastid | artistid |   podcastname
-----------+----------+------------------
 PO001     | AR002    | Infulencers
 PO002     | AR003    | Technical Sapien
 PO003     | AR005    | The Ranveer Show
 PO004     | AR002    | The Daily
(4 rows)
```

5.2.10 Recording :



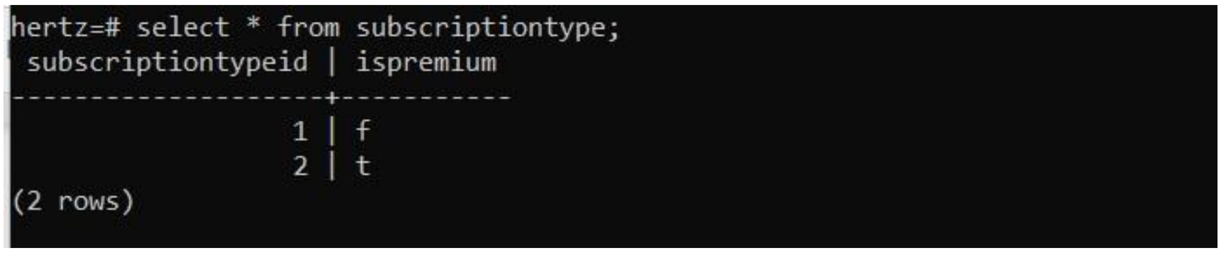
```
hertz=# select * from recording;
 recordingid | recordingname |     duration      | podcastid | songid |   genre
-------------+---------------+-------------------+-----------+--------+-----------
 RE004       | Recording4    | 19 days 01:57:29 | PO002     | SO004  | pop
 RE002       | Recording2    | 10 days 07:36:26 | PO004     | SO001  | rock
 RE003       | Recording3    | 6 days 12:43:53  | PO003     | SO003  | hip hop
 RE001       | Recording1    | 15 days 10:21:59 | PO002     | SO004  | classical
 RE005       | Recording5    | 25 days 13:39:31 | PO001     | SO002  | classical
(5 rows)
```

5.2.11 Songs :



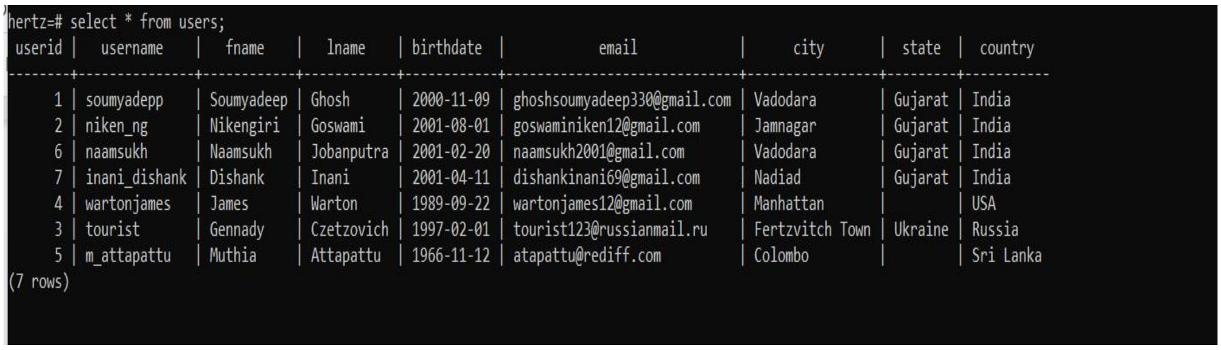
```
hertz=# select * from song;
 songid |    songname     |  songdate
--------+-----------------+------------
 S0001  | Razor           | 2020-06-05
 S0004  | cold            | 2020-05-04
 S0005  | Save your tears | 2019-02-02
 S0003  | Thousand years  | 2020-05-02
 S0002  | Day dreaming    | 2020-11-12
(5 rows)
```

5.2.12 SubscriptionType :



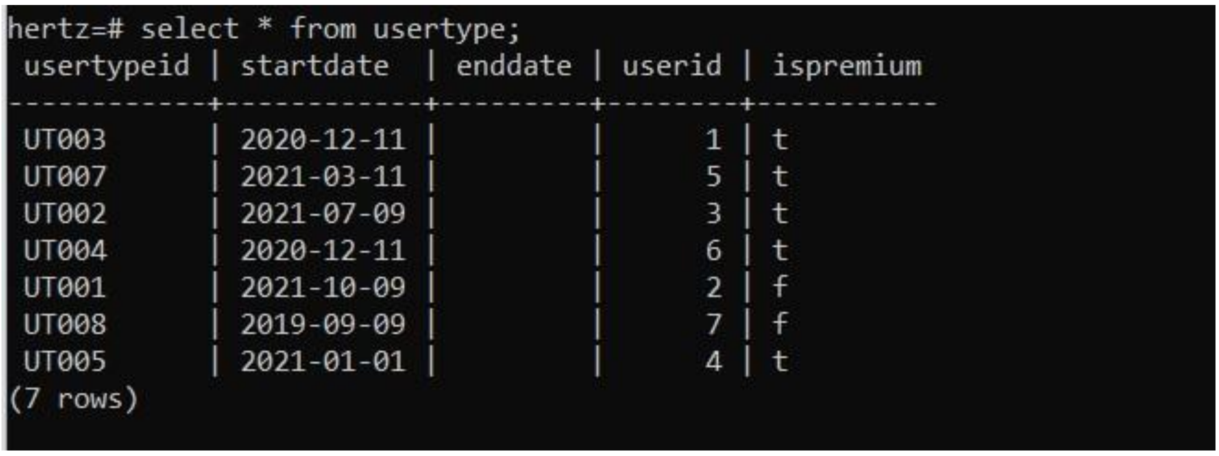
```
hertz=# select * from subscriptiontype;
 subscriptiontypeid | ispremium
--------------------+-----------
                  1 | f
                  2 | t
(2 rows)
```

5.2.13 Users :



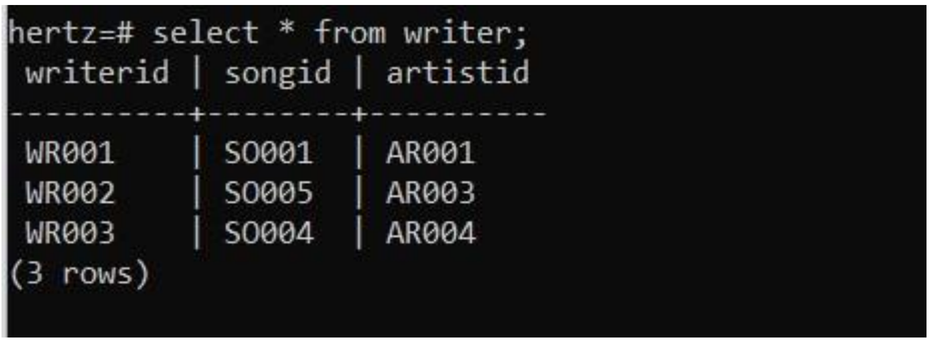
```
hertz=# select * from users;
 userid |   username    |   fname    |   lname    | birthdate  |            email             |      city       |  state  |  country
--------+---------------+------------+------------+------------+------------------------------+-----------------+---------+-----------
      1 | soumyadepp    | Soumyadeep | Ghosh      | 2000-11-09 | ghoshsoumyadeep330@gmail.com | Vadodara        | Gujarat | India
      2 | niken_ng      | Nikengiri  | Goswami    | 2001-08-01 | goswaminiken12@gmail.com     | Jamnagar        | Gujarat | India
      6 | naamsukh      | Naamsukh   | Jobanputra | 2001-02-20 | naamsukh2001@gmail.com       | Vadodara        | Gujarat | India
      7 | inani_dishank | Dishank    | Inani      | 2001-04-11 | dishankinani69@gmail.com     | Nadiad          | Gujarat | India
      4 | wartonjames   | James      | Warton     | 1989-09-22 | wartonjames12@gmail.com      | Manhattan       |         | USA
      3 | tourist       | Gennady    | Czetzovich | 1997-02-01 | tourist123@russianmail.ru    | Fertzvitch Town | Ukraine | Russia
      5 | m_attapattu   | Muthia     | Attapattu  | 1966-11-12 | atapattu@rediff.com          | Colombo         |         | Sri Lanka
(7 rows)
```

5.2.14 User type :



```
hertz=# select * from usertype;
 usertypeid | startdate  | enddate | userid | ispremium
------------+------------+---------+--------+-----------
 UT003      | 2020-12-11 |         |      1 | t
 UT007      | 2021-03-11 |         |      5 | t
 UT002      | 2021-07-09 |         |      3 | t
 UT004      | 2020-12-11 |         |      6 | t
 UT001      | 2021-10-09 |         |      2 | f
 UT008      | 2019-09-09 |         |      7 | f
 UT005      | 2021-01-01 |         |      4 | t
(7 rows)
```

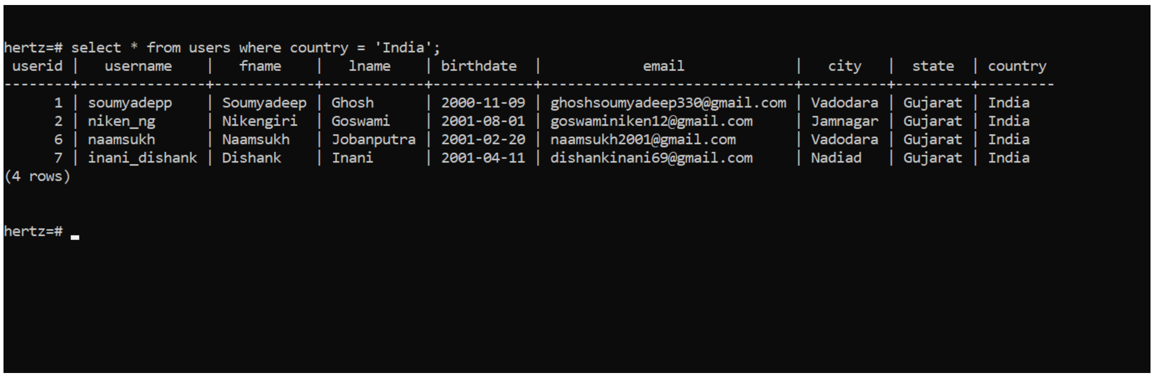
5.2.15 Writer :



```
hertz=# select * from writer;
 writerid | songid | artistid
----------+--------+----------
 WR001    | SO001  | AR001
 WR002    | SO005  | AR003
 WR003    | SO004  | AR004
(3 rows)
```

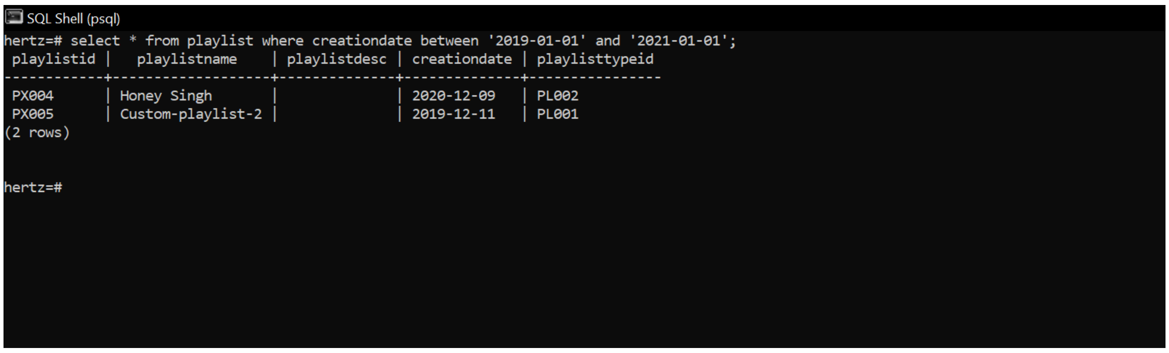
5.3 QUERIES USING BASIC DBMS CONSTRUCTS:

5.3.1 Display the users who are from India.



```
hertz=# select * from users where country = 'India';
 userid |   username    |   fname    |   lname    | birthdate  |            email             |   city   |  state  | country
--------+---------------+------------+------------+------------+------------------------------+----------+---------+---------
      1 | soumyadepp    | Soumyadeep | Ghosh      | 2000-11-09 | ghoshsoumyadeep330@gmail.com | Vadodara | Gujarat | India
      2 | niken_ng      | Nikengiri  | Goswami    | 2001-08-01 | goswaminiken12@gmail.com     | Jamnagar | Gujarat | India
      6 | naamsukh      | Naamsukh   | Jobanputra | 2001-02-20 | naamsukh2001@gmail.com       | Vadodara | Gujarat | India
      7 | inani_dishank | Dishank    | Inani      | 2001-04-11 | dishankinani69@gmail.com     | Nadiad   | Gujarat | India
(4 rows)


hertz=#
```

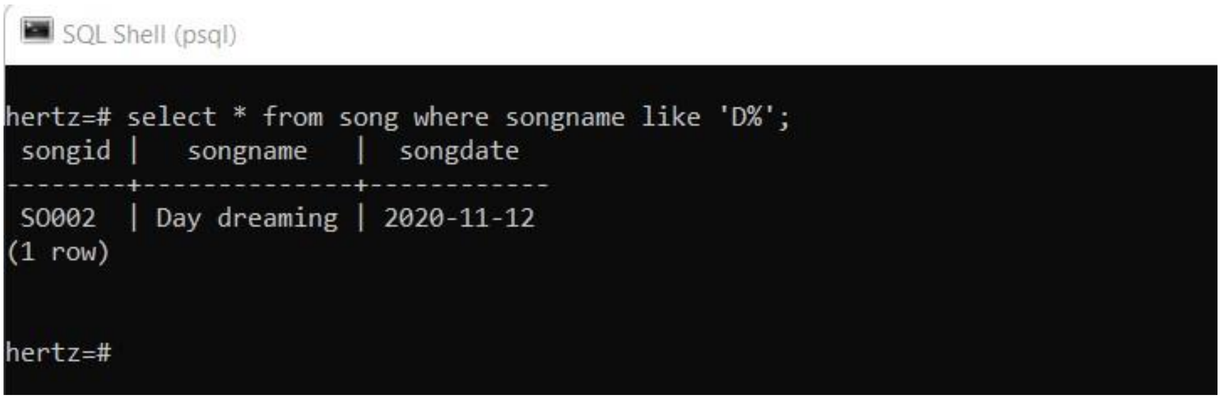
5.3.2 Display the playlist which is created between year 2019-2021



SQL Shell (psql)

```
hertz=# select * from playlist where creationdate between '2019-01-01' and '2021-01-01';
 playlistid |   playlistname    | playlistdesc | creationdate | playlisttypeid
------------+-------------------+--------------+--------------+----------------
 PX004      | Honey Singh       |              | 2020-12-09   | PL002
 PX005      | Custom-playlist-2 |              | 2019-12-11   | PL001
(2 rows)


hertz=#
```

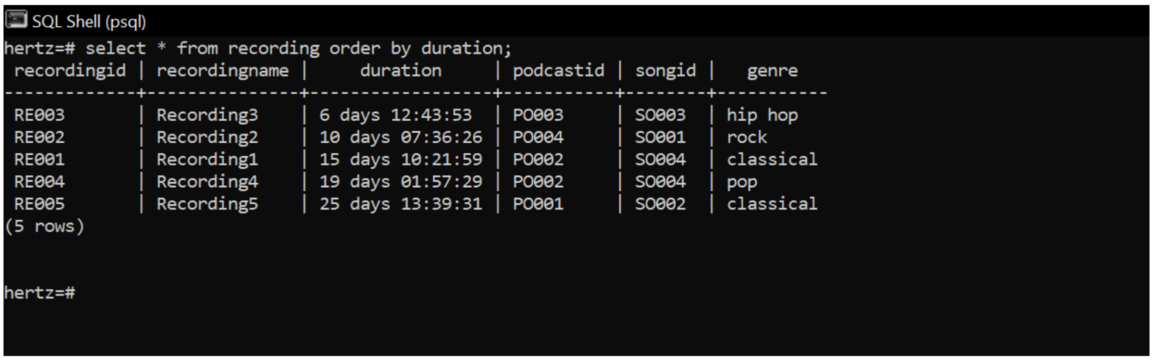
5.3.3 Display the songs whose name start with D.



SQL Shell (psql)

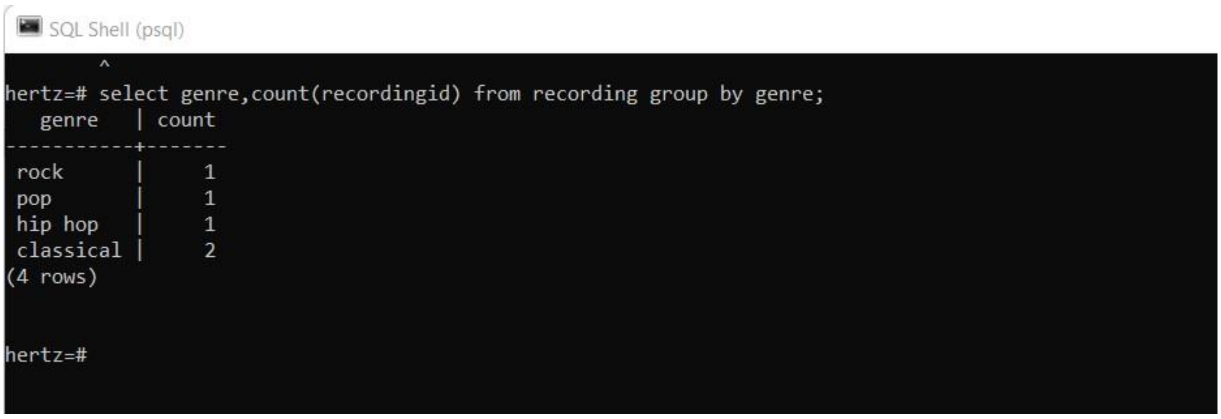
```
hertz=# select * from song where songname like 'D%';
 songid |   songname    |  songdate
--------+---------------+------------
 SO002  | Day dreaming  | 2020-11-12
(1 row)


hertz=#
```

5.3.4 Display the recording table and sort by duration.



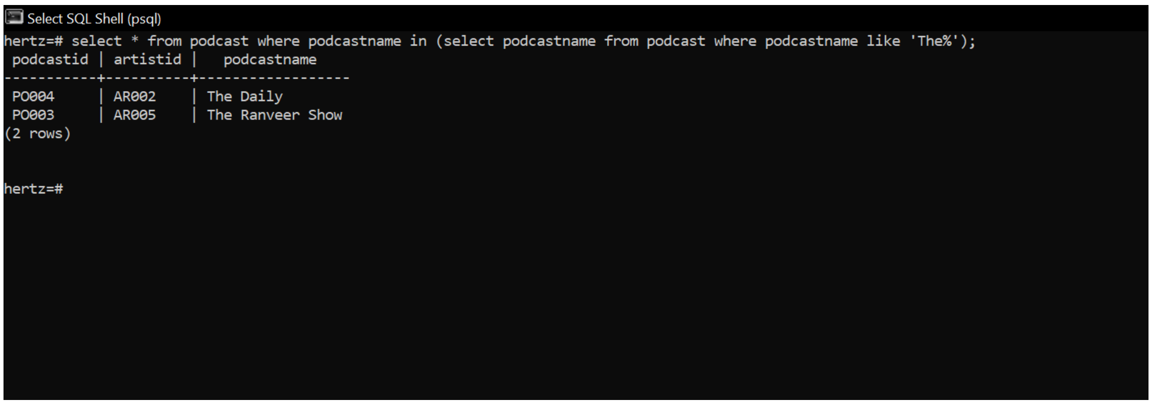
```
SQL Shell (psql)
hertz=# select * from recording order by duration;
 recordingid | recordingname |      duration      | podcastid | songid |   genre
-------------+---------------+--------------------+-----------+--------+-----------
 RE003       | Recording3    | 6 days 12:43:53    | PO003     | SO003  | hip hop
 RE002       | Recording2    | 10 days 07:36:26   | PO004     | SO001  | rock
 RE001       | Recording1    | 15 days 10:21:59   | PO002     | SO004  | classical
 RE004       | Recording4    | 19 days 01:57:29   | PO002     | SO004  | pop
 RE005       | Recording5    | 25 days 13:39:31   | PO001     | SO002  | classical
(5 rows)


hertz=#
```

5.3.5 Display the count of all the recordings by genre.



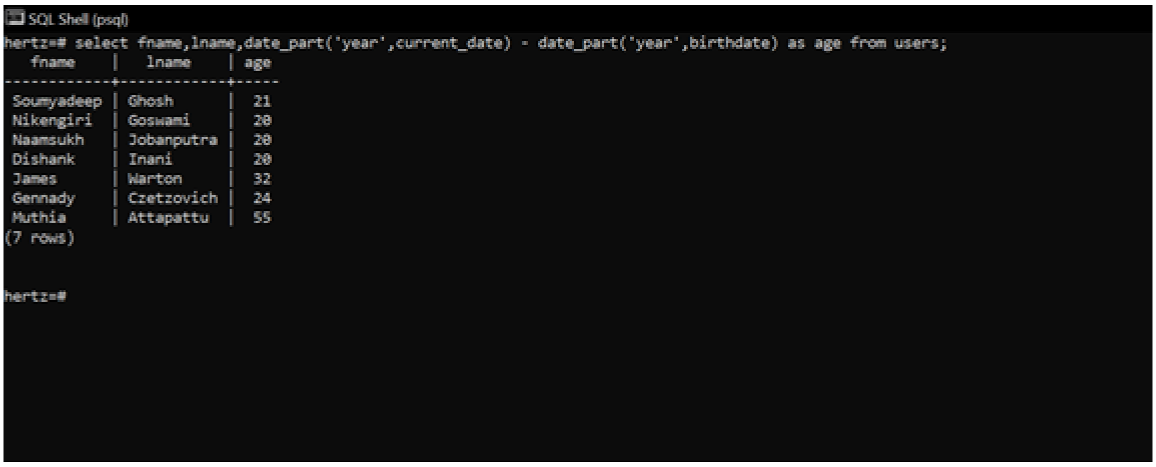
```
SQL Shell (psql)
        ^
hertz=# select genre,count(recordingid) from recording group by genre;
   genre   | count
-----------+-------
 rock      |     1
 pop       |     1
 hip hop   |     1
 classical |     2
(4 rows)


hertz=#
```

5.3.6 Display the podcasts which starts with ‘THE’



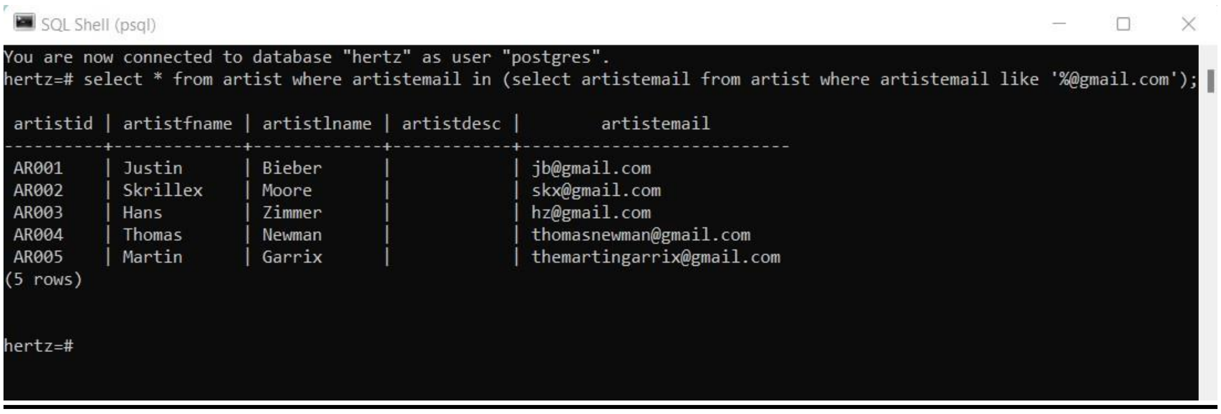
```
Select SQL Shell (psql)
hertz=# select * from podcast where podcastname in (select podcastname from podcast where podcastname like 'The%');
 podcastid | artistid |   podcastname
-----------+----------+-----------------
 PO004     | AR002    | The Daily
 PO003     | AR005    | The Ranveer Show
(2 rows)


hertz=#
```

5.3.7 Display the fname,lname of users with their age in ascending order.



```
SQL Shell (psql)
hertz=# select fname,lname,date_part('year',current_date) - date_part('year',birthdate) as age from users;
   fname    |   lname    | age
------------+------------+-----
 Soumyadeep | Ghosh      |  21
 Nikengiri  | Goswami    |  20
 Naamsukh   | Jobanputra |  20
 Dishank    | Inani      |  20
 James      | Warton     |  32
 Gennady    | Czetzovich |  24
 Muthia     | Attapattu  |  55
(7 rows)


hertz=#
```

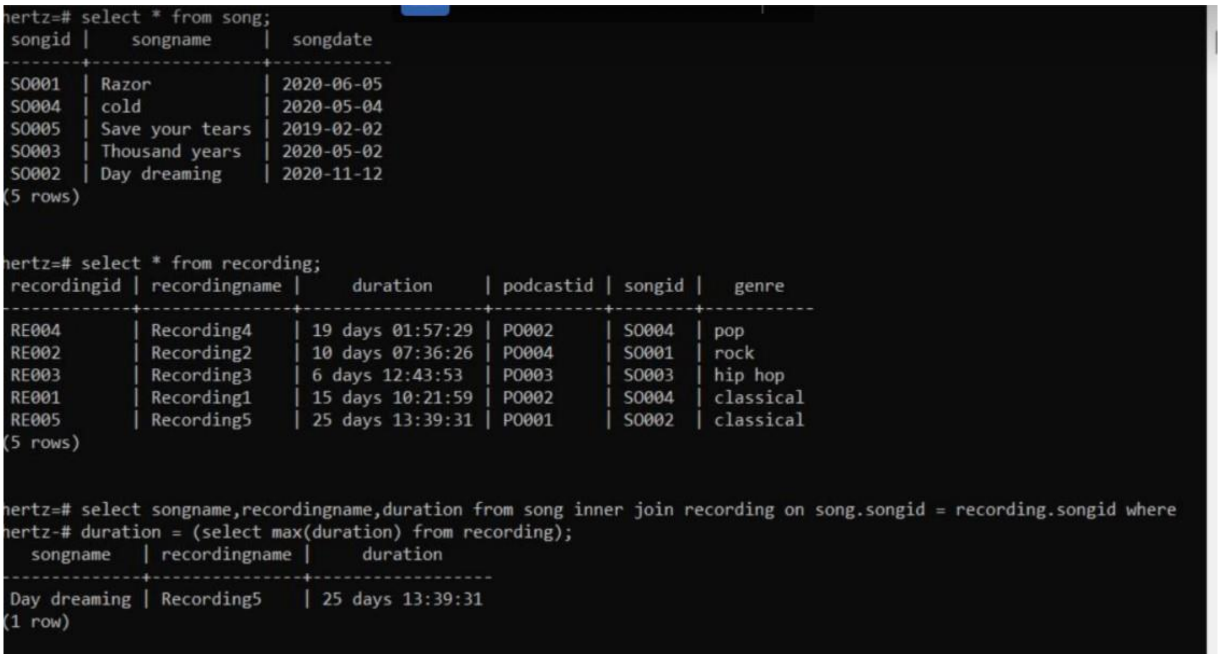
5.3.8 Display all the artist who has a gmail account.



```
SQL Shell (psql)
You are now connected to database "hertz" as user "postgres".
hertz=# select * from artist where artistemail in (select artistemail from artist where artistemail like '%@gmail.com');

 artistid | artistfname | artistlname | artistdesc |        artistemail
----------+-------------+-------------+------------+---------------------------
 AR001    | Justin      | Bieber      |            | jb@gmail.com
 AR002    | Skrillex    | Moore       |            | skx@gmail.com
 AR003    | Hans        | Zimmer      |            | hz@gmail.com
 AR004    | Thomas      | Newman      |            | thomasnewman@gmail.com
 AR005    | Martin      | Garrix      |            | themartingarrix@gmail.com
(5 rows)


hertz=#
```

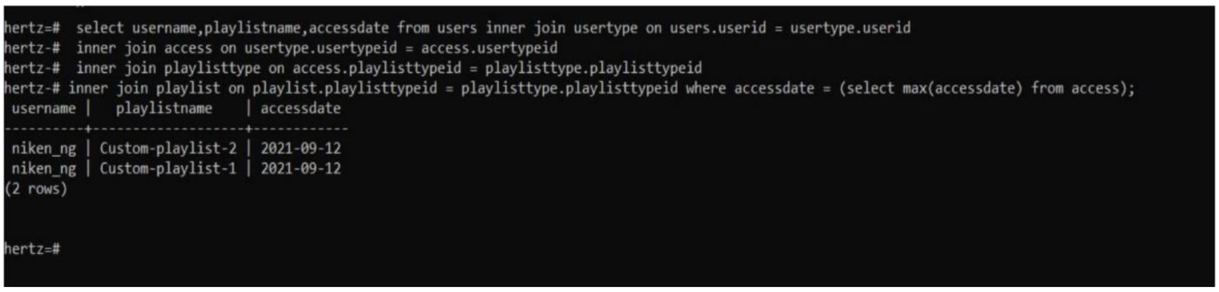
5.4 QUERIES USING JOIN AND SUBQUERIES

5.4.1 Write a query to find the song name with maximum duration of recording



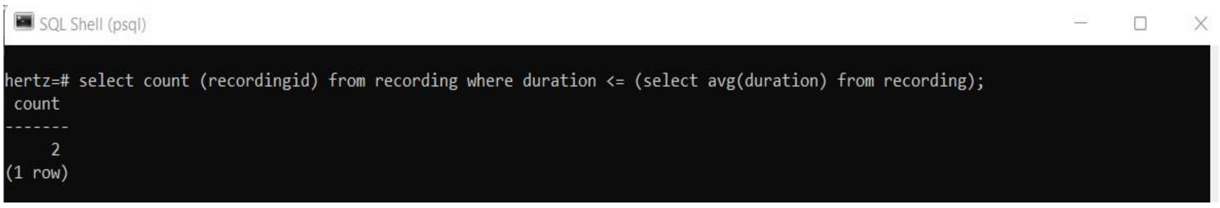
```
ertz=# select * from song;
 songid |    songname      |  songdate
--------+------------------+------------
 S0001  | Razor            | 2020-06-05
 S0004  | cold             | 2020-05-04
 S0005  | Save your tears  | 2019-02-02
 S0003  | Thousand years   | 2020-05-02
 S0002  | Day dreaming     | 2020-11-12
(5 rows)


ertz=# select * from recording;
 recordingid | recordingname |     duration     | podcastid | songid |   genre
-------------+---------------+------------------+-----------+--------+-----------
 RE004       | Recording4    | 19 days 01:57:29 | P0002     | S0004  | pop
 RE002       | Recording2    | 10 days 07:36:26 | P0004     | S0001  | rock
 RE003       | Recording3    | 6 days 12:43:53  | P0003     | S0003  | hip hop
 RE001       | Recording1    | 15 days 10:21:59 | P0002     | S0004  | classical
 RE005       | Recording5    | 25 days 13:39:31 | P0001     | S0002  | classical
(5 rows)


ertz=# select songname,recordingname,duration from song inner join recording on song.songid = recording.songid where
ertz-# duration = (select max(duration) from recording);
   songname    | recordingname |     duration
---------------+---------------+------------------
 Day dreaming  | Recording5    | 25 days 13:39:31
(1 row)
```

5.4.2 Write a query which finds out the playlist which is been accessed recently by the users



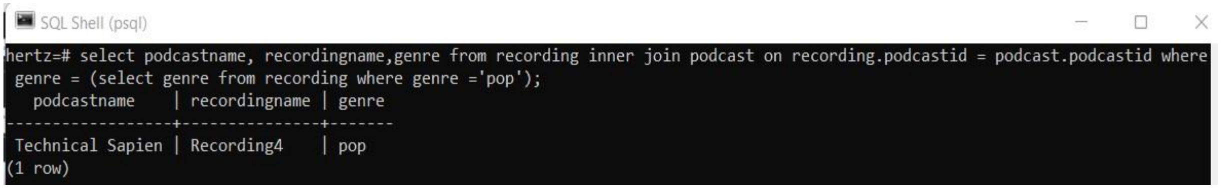
```
hertz=#  select username,playlistname,accessdate from users inner join usertype on users.userid = usertype.userid
hertz-#  inner join access on usertype.usertypeid = access.usertypeid
hertz-#  inner join playlisttype on access.playlisttypeid = playlisttype.playlisttypeid
hertz-# inner join playlist on playlist.playlisttypeid = playlisttype.playlisttypeid where accessdate = (select max(accessdate) from access);
 username |   playlistname    | accessdate
----------+-------------------+------------
 niken_ng | Custom-playlist-2 | 2021-09-12
 niken_ng | Custom-playlist-1 | 2021-09-12
(2 rows)


hertz=#
```

5.4.3. Write a query that finds the recording which is having a duration less than equal to the average duration.



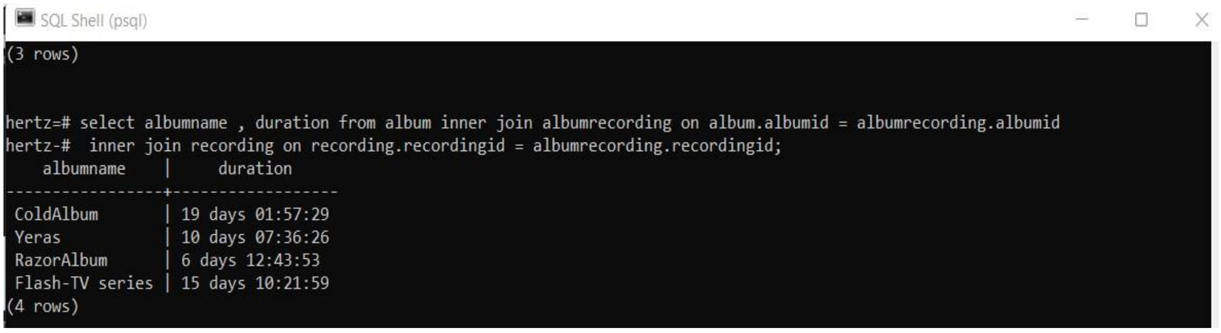
```
hertz=# select count (recordingid) from recording where duration <= (select avg(duration) from recording);
 count
-------
     2
(1 row)
```

5.4.4 Write a query which finds the podcast which is having a genre='pop'.



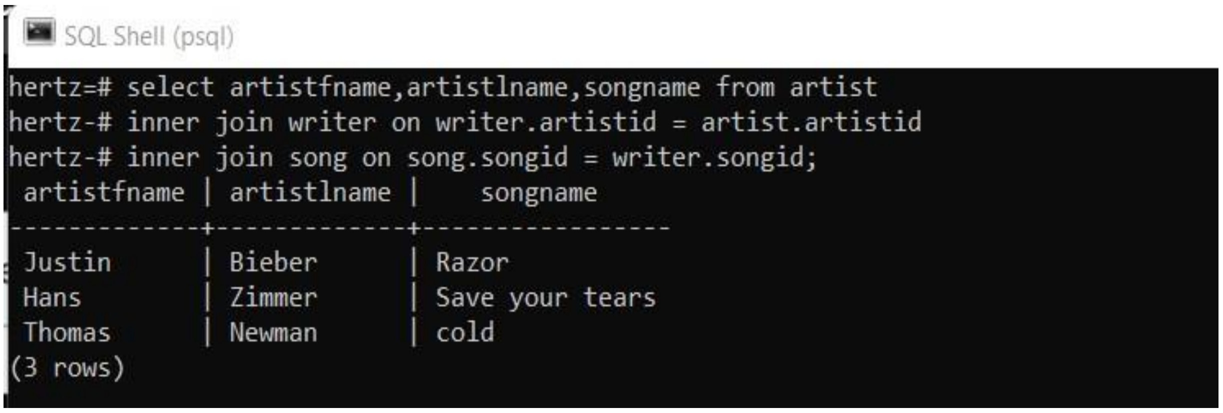
```
hertz=# select podcastname, recordingname,genre from recording inner join podcast on recording.podcastid = podcast.podcastid where
 genre = (select genre from recording where genre ='pop');
   podcastname    | recordingname | genre
------------------+---------------+-------
 Technical Sapien | Recording4    | pop
(1 row)
```

5.4.5 Write a query which finds the album and its duration.



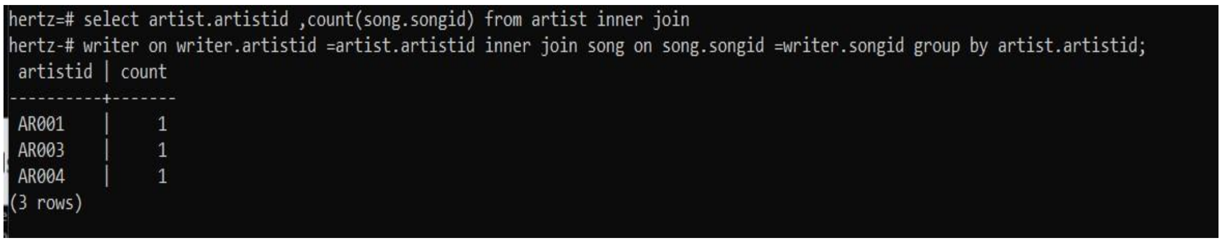
```
(3 rows)


hertz=# select albumname , duration from album inner join albumrecording on album.albumid = albumrecording.albumid
hertz-#  inner join recording on recording.recordingid = albumrecording.recordingid;
    albumname    |     duration
-----------------+------------------
 ColdAlbum       | 19 days 01:57:29
 Yeras           | 10 days 07:36:26
 RazorAlbum      | 6 days 12:43:53
 Flash-TV series | 15 days 10:21:59
(4 rows)
```

5.4.6 Writer a query which finds the artist and the song written by him/her.



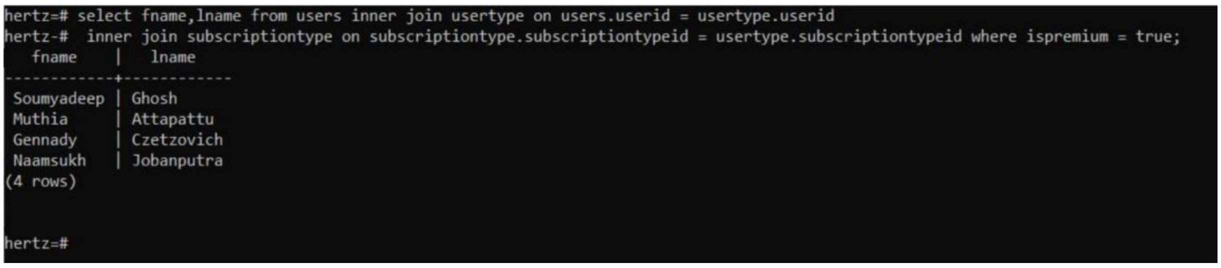
```
hertz=# select artistfname,artistlname,songname from artist
hertz-# inner join writer on writer.artistid = artist.artistid
hertz-# inner join song on song.songid = writer.songid;
 artistfname | artistlname |    songname
-------------+-------------+-----------------
 Justin      | Bieber      | Razor
 Hans        | Zimmer      | Save your tears
 Thomas      | Newman      | cold
(3 rows)
```

5.4.7 Write a query that finds the number of songs the artist has written.



```
hertz=# select artist.artistid ,count(song.songid) from artist inner join
hertz-# writer on writer.artistid =artist.artistid inner join song on song.songid =writer.songid group by artist.artistid;
 artistid | count
----------+-------
 AR001    |     1
 AR003    |     1
 AR004    |     1
(3 rows)
```

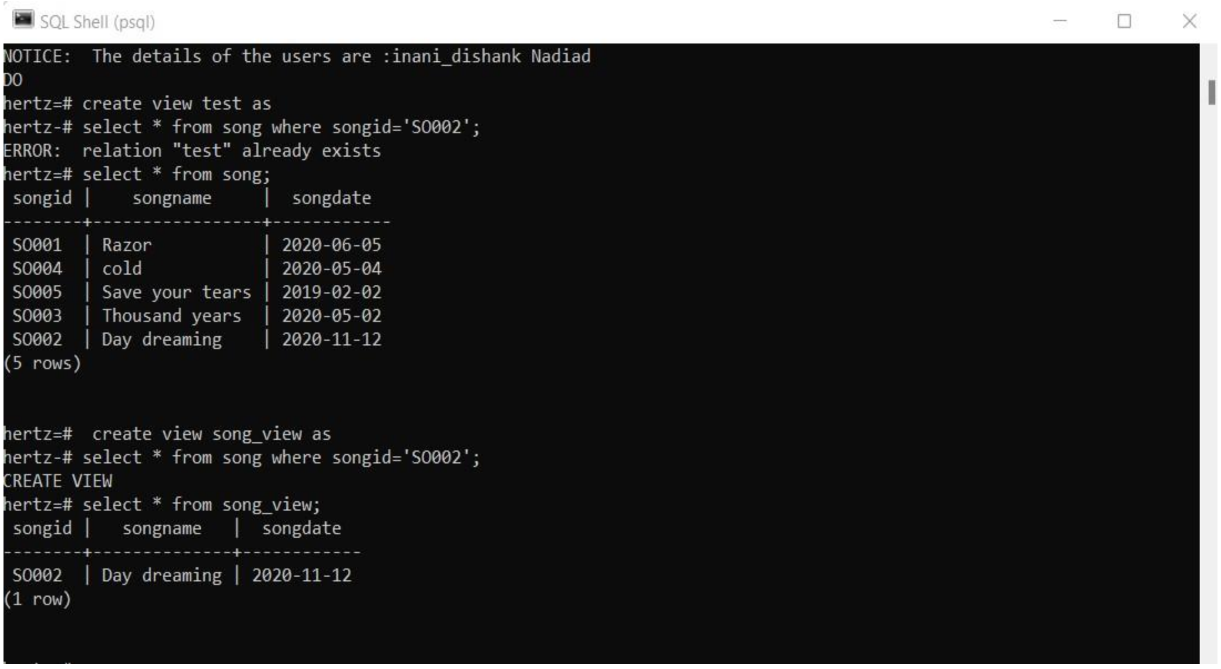
5.4.8 Write a query that finds the users who have taken premium



```
hertz=# select fname,lname from users inner join usertype on users.userid = usertype.userid
hertz-#  inner join subscriptiontype on subscriptiontype.subscriptiontypeid = usertype.subscriptiontypeid where ispremium = true;
   fname    |   lname
------------+------------
 Soumyadeep | Ghosh
 Muthia     | Attapattu
 Gennady    | Czetzovich
 Naamsukh   | Jobanputra
(4 rows)


hertz=#
```

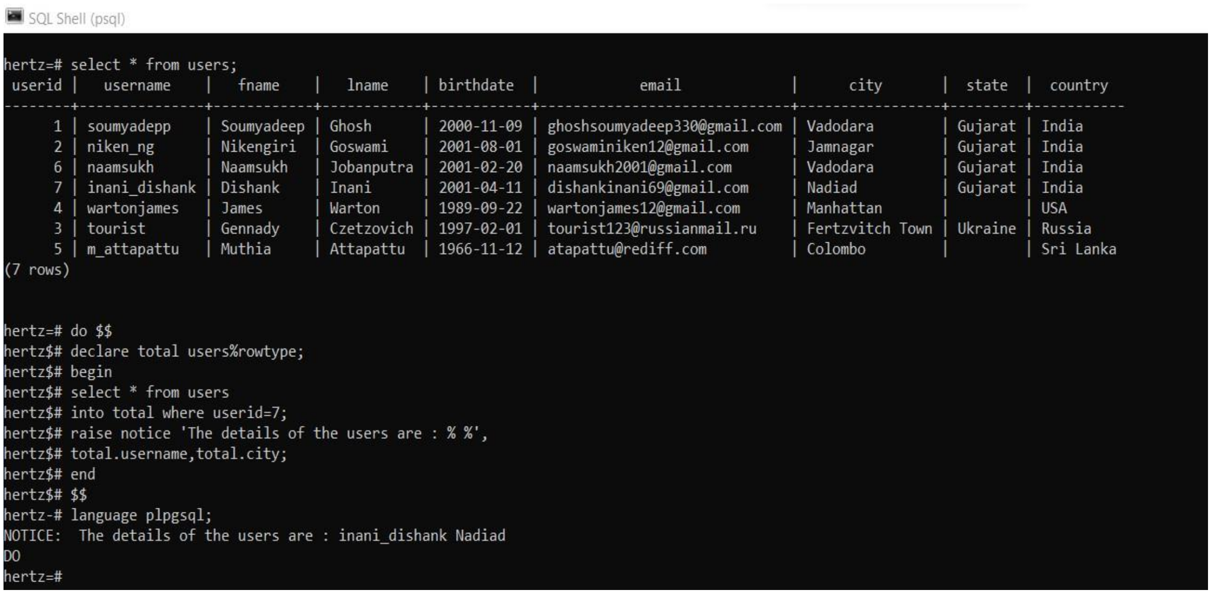
5.5 PL/SQL:

VIEW



```
NOTICE:  The details of the users are :inani_dishank Nadiad
DO
hertz=# create view test as
hertz-# select * from song where songid='S0002';
ERROR:  relation "test" already exists
hertz=# select * from song;
 songid |    songname      |  songdate
--------+-----------------+------------
 S0001  | Razor           | 2020-06-05
 S0004  | cold            | 2020-05-04
 S0005  | Save your tears | 2019-02-02
 S0003  | Thousand years  | 2020-05-02
 S0002  | Day dreaming    | 2020-11-12
(5 rows)


hertz=#  create view song_view as
hertz-# select * from song where songid='S0002';
CREATE VIEW
hertz=# select * from song_view;
 songid |   songname   |  songdate
--------+--------------+------------
 S0002  | Day dreaming | 2020-11-12
(1 row)
```

ROWTYPE



```
hertz=# select * from users;
 userid |   username    |   fname    |   lname    | birthdate  |             email             |      city       |  state  |  country
--------+---------------+------------+------------+------------+-------------------------------+-----------------+---------+-----------
      1 | soumyadepp    | Soumyadeep | Ghosh      | 2000-11-09 | ghoshsoumyadeep330@gmail.com  | Vadodara        | Gujarat | India
      2 | niken_ng      | Nikengiri  | Goswami    | 2001-08-01 | goswaminiken12@gmail.com      | Jamnagar        | Gujarat | India
      6 | naamsukh      | Naamsukh   | Jobanputra | 2001-02-20 | naamsukh2001@gmail.com        | Vadodara        | Gujarat | India
      7 | inani_dishank | Dishank    | Inani      | 2001-04-11 | dishankinani69@gmail.com      | Nadiad          | Gujarat | India
      4 | wartonjames   | James      | Warton     | 1989-09-22 | wartonjames12@gmail.com       | Manhattan       |         | USA
      3 | tourist       | Gennady    | Czetzovich | 1997-02-01 | tourist123@russianmail.ru     | Fertzvitch Town | Ukraine | Russia
      5 | m_attapattu   | Muthia     | Attapattu  | 1966-11-12 | atapattu@rediff.com           | Colombo         |         | Sri Lanka
(7 rows)


hertz=# do $$
hertz$# declare total users%rowtype;
hertz$# begin
hertz$# select * from users
hertz$# into total where userid=7;
hertz$# raise notice 'The details of the users are : % %',
hertz$# total.username,total.city;
hertz$# end
hertz$# $$
hertz-# language plpgsql;
NOTICE:  The details of the users are : inani_dishank Nadiad
DO
hertz=#
```

5.6 FUNCTION & TRIGGERS:

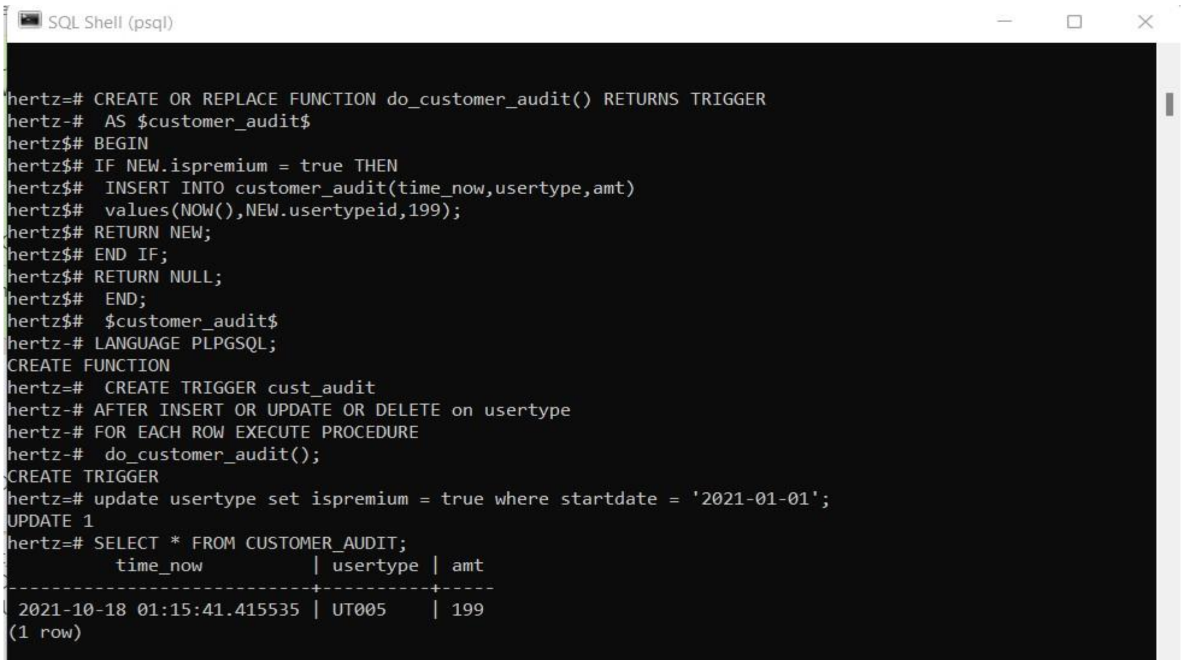
5.6.1 Function and trigger which keeps log of all the users who bought premium and make an entry in the customer_audit table .

Function:

```
CREATE OR REPLACE FUNCTION do_customer_audit() RETURNS TRIGGER
AS $customer_audit$
BEGIN
IF NEW.ispremium = true THEN
INSERT INTO customer_audit(time_now,usertype,amt)
values(NOW(),NEW.usertypeid,199);
RETURN NEW;
END IF;
RETURN NULL;
END;
$customer_audit$
LANGUAGE PLPGSQL;
```

Trigger:

```
CREATE TRIGGER cust_audit
AFTER INSERT OR UPDATE OR DELETE on usertype
FOR EACH ROW EXECUTE PROCEDURE
do_customer_audit();
```



```
hertz=# CREATE OR REPLACE FUNCTION do_customer_audit() RETURNS TRIGGER
hertz-#  AS $customer_audit$
hertz$# BEGIN
hertz$# IF NEW.ispremium = true THEN
hertz$#  INSERT INTO customer_audit(time_now,usertype,amt)
hertz$#  values(NOW(),NEW.usertypeid,199);
hertz$# RETURN NEW;
hertz$# END IF;
hertz$# RETURN NULL;
hertz$#  END;
hertz$#  $customer_audit$
hertz-# LANGUAGE PLPGSQL;
CREATE FUNCTION
hertz=#  CREATE TRIGGER cust_audit
hertz-# AFTER INSERT OR UPDATE OR DELETE on usertype
hertz-# FOR EACH ROW EXECUTE PROCEDURE
hertz-#  do_customer_audit();
CREATE TRIGGER
hertz=# update usertype set ispremium = true where startdate = '2021-01-01';
UPDATE 1
hertz=# SELECT * FROM CUSTOMER_AUDIT;
          time_now          | usertype | amt
----------------------------+----------+-----
 2021-10-18 01:15:41.415535 | UT005    | 199
(1 row)
```

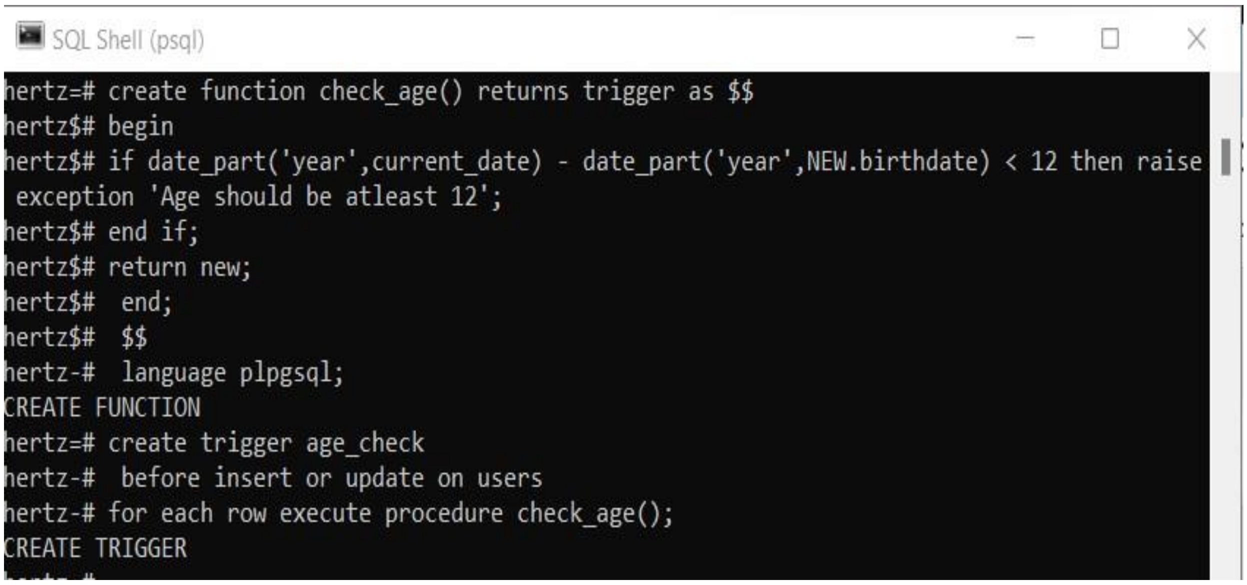
5.6.2 Function and trigger which checks user’s age and if it is less than 12 then it won’t allow to insert or update in user’s table.

FUNCTION :

```
CREATE FUNCTION CHECK_AGE() RETURNS TRIGGER AS $$
BEGIN
IF DATE_PART('YEAR',CURRENT_DATE) -
DATE_PART('YEAR',NEW.BIRTHDATE) < 12 THEN RAISE EXCEPTION 'AGE
SHOULD BE ATLEAST 12';
END IF;
RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;
```

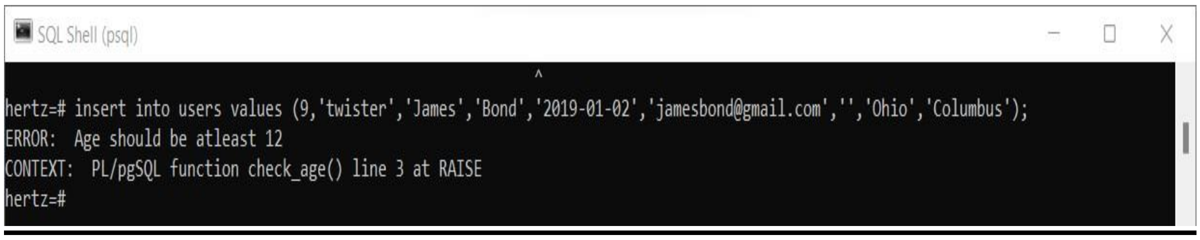
TRIGGER :

```
CREATE TRIGGER AGE_CHECK
BEFORE INSERT OR UPDATE ON USERS
FOR EACH ROW EXECUTE PROCEDURE CHECK_AGE();
```



SQL Shell (psql)

```
hertz=# create function check_age() returns trigger as $$
hertz$# begin
hertz$# if date_part('year',current_date) - date_part('year',NEW.birthdate) < 12 then raise
 exception 'Age should be atleast 12';
hertz$# end if;
hertz$# return new;
hertz$#  end;
hertz$#  $$
hertz-#  language plpgsql;
CREATE FUNCTION
hertz=# create trigger age_check
hertz-#  before insert or update on users
hertz-# for each row execute procedure check_age();
CREATE TRIGGER
```



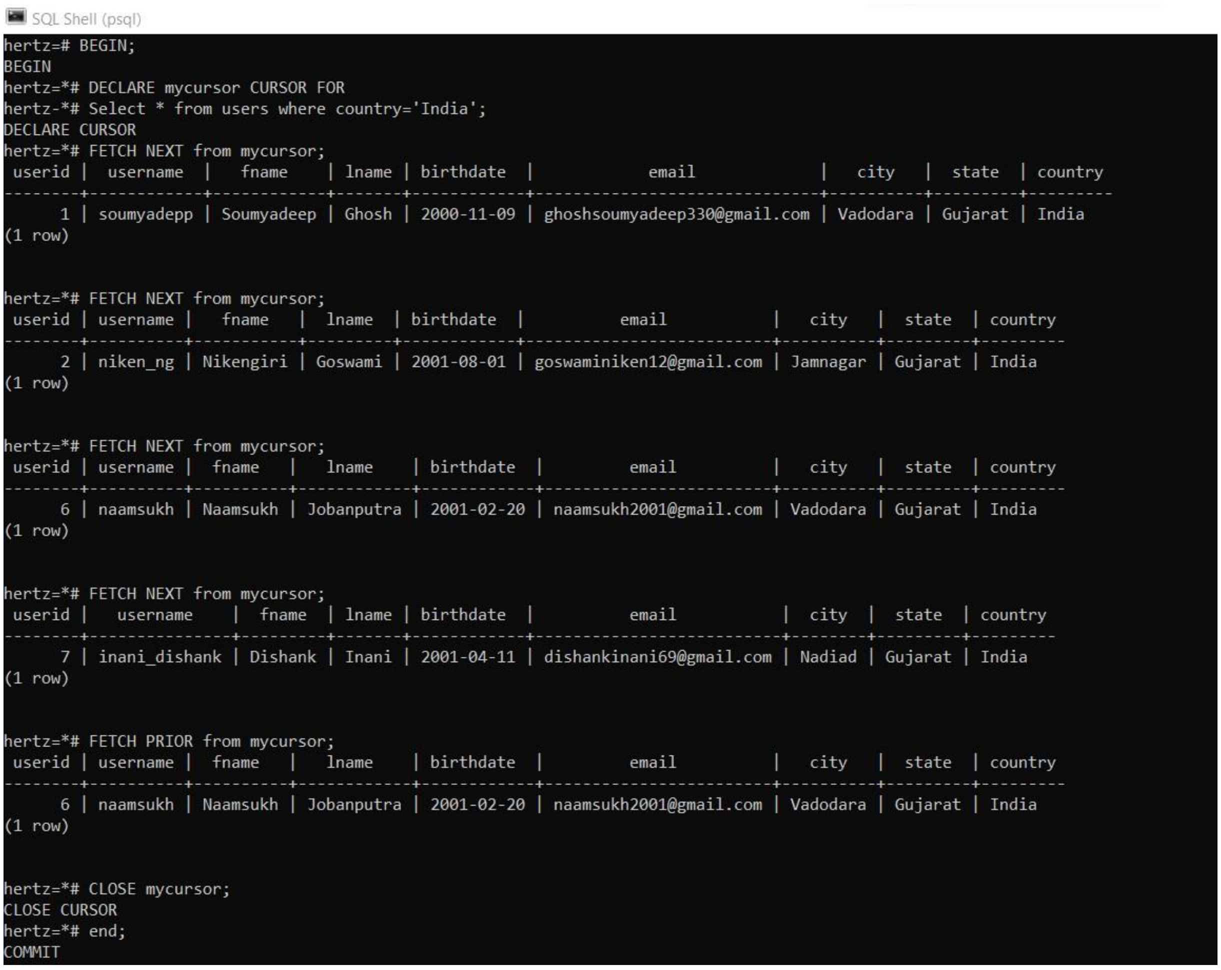
SQL Shell (psql)

```
                                                            ^
hertz=# insert into users values (9,'twister','James','Bond','2019-01-02','jamesbond@gmail.com','','Ohio','Columbus');
ERROR:  Age should be atleast 12
CONTEXT:  PL/pgSQL function check_age() line 3 at RAISE
hertz=#
```

5.7 CURSOR:

Create a Cursor which traverses through users table where the country of the user is INDIA.

```
BEGIN
DECLARE mycursor CURSOR FOR
SELECT * FROM USERS WHERE COUNTRY = 'India';

FETCH NEXT FROM mycursor;

FETCH PRIOR FROM mycursor;
CLOSE mycursor;
end;
```


SQL Shell (psql)

```
hertz=# BEGIN;
BEGIN
hertz=*# DECLARE mycursor CURSOR FOR
hertz-*# Select * from users where country='India';
DECLARE CURSOR
hertz=*# FETCH NEXT from mycursor;
 userid |  username  |   fname    | lname | birthdate  |            email             |   city   |  state  | country
--------+------------+------------+-------+------------+------------------------------+----------+---------+---------
      1 | soumyadepp | Soumyadeep | Ghosh | 2000-11-09 | ghoshsoumyadeep330@gmail.com | Vadodara | Gujarat | India
(1 row)


hertz=*# FETCH NEXT from mycursor;
 userid | username |   fname   |  lname  | birthdate  |          email           |   city   |  state  | country
--------+----------+-----------+---------+------------+--------------------------+----------+---------+---------
      2 | niken_ng | Nikengiri | Goswami | 2001-08-01 | goswaminiken12@gmail.com | Jamnagar | Gujarat | India
(1 row)


hertz=*# FETCH NEXT from mycursor;
 userid | username |  fname   |   lname    | birthdate  |         email          |   city   |  state  | country
--------+----------+----------+------------+------------+------------------------+----------+---------+---------
      6 | naamsukh | Naamsukh | Jobanputra | 2001-02-20 | naamsukh2001@gmail.com | Vadodara | Gujarat | India
(1 row)


hertz=*# FETCH NEXT from mycursor;
 userid |   username    |  fname  | lname | birthdate  |          email           |  city  |  state  | country
--------+---------------+---------+-------+------------+--------------------------+--------+---------+---------
      7 | inani_dishank | Dishank | Inani | 2001-04-11 | dishankinani69@gmail.com | Nadiad | Gujarat | India
(1 row)


hertz=*# FETCH PRIOR from mycursor;
 userid | username |  fname   |   lname    | birthdate  |         email          |   city   |  state  | country
--------+----------+----------+------------+------------+------------------------+----------+---------+---------
      6 | naamsukh | Naamsukh | Jobanputra | 2001-02-20 | naamsukh2001@gmail.com | Vadodara | Gujarat | India
(1 row)


hertz=*# CLOSE mycursor;
CLOSE CURSOR
hertz=*# end;
COMMIT
```

6. FUTURE ENHANCEMENTS OF THE SYSTEM

- We will design Front-end using React Framework and Develop Back-end in NodeJS
- Methods and user data input will be a lot easy after the implementation of GUI.
- In the future, we can place the system on the cloud so the maintenance of the data can be reduced.

7. BIBLIOGRAPHY

- We created ER-Model on Whimsical and Relational Schema on Creately
- ER-MODEL - **https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD**
- RELATIONAL SCHEMA - **https://app.creately.com/diagram/yQAR0D8Dgpa/edit**
- For the implementation of this project, we referred to materials shared by Prof. Archana N. Vyas and the following websites and books:

Book:

Database System Concepts
-Henry F. Korth & A. Silberschatz 2nd Ed. McGraw-Hill 1991

Websites:

- https://www.w3schools.com/sql/sql_syntax.asp
- https://www.tutorialspoint.com/
- https://dev.mysql.com/doc/
- https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/